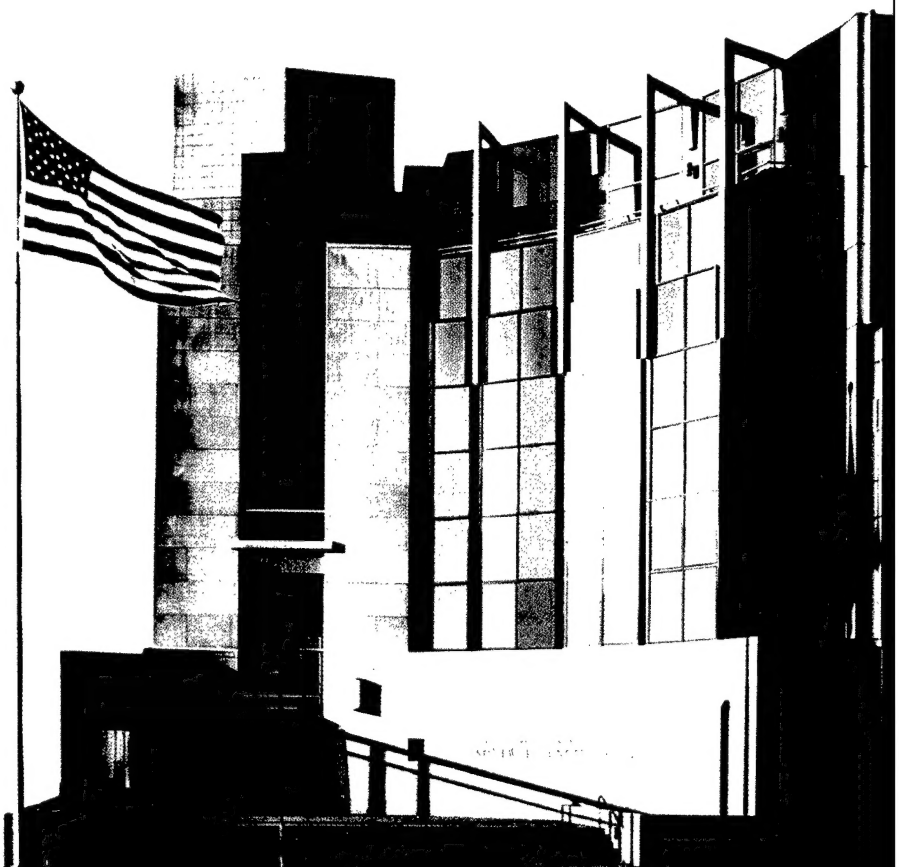
Carnegie Mellon
Software Engineering Institute

Perceived Control of Software Developers and Its Impact on the Successful Diffusion of Information Technology

Gina Green, Baylor University
Alan R. Hevner, University of South Florida

April 1999



SPECIAL REPORT
CMU/SEI-98-SR-013

Carnegie Mellon University does not discriminate and Carnegie Mellon University is required not to discriminate in admission, employment, or administration of its programs or activities on the basis of race, color, national origin, sex or handicap in violation of Title VI of the Civil Rights Act of 1964, Title IX of the Educational Amendments of 1972 and Section 504 of the Rehabilitation Act of 1973 or other federal, state, or local laws or executive orders.

In addition, Carnegie Mellon University does not discriminate in admission, employment or administration of its programs on the basis of religion, creed, ancestry, belief, age, veteran status, sexual orientation or in violation of federal, state, or local laws or executive orders. However, in the judgment of the Carnegie Mellon Human Relations Commission, the Department of Defense policy of "Don't ask, don't tell, don't pursue" excludes openly gay, lesbian and bisexual students from receiving ROTC scholarships or serving in the military. Nevertheless, all ROTC classes at Carnegie Mellon University are available to all students.

Inquiries concerning application of these statements should be directed to the Provost, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-6684 or the Vice President for Enrollment, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-2056.

Obtain general information about Carnegie Mellon University by calling (412) 268-2000.

This work is sponsored by the U.S. Department of Defense and by a Cooperative Research and Development Agreement with the University of South Florida and Baylor University.

The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number F19628-95-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 52.227-7013.

This document is available through Asset Source for Software Engineering Technology (ASSET): 1350 Earl L. Core Road; PO Box 3305; Morgantown, West Virginia 26505 / Phone: (304) 284-9000 or toll-free in the U.S. 1-800-547-8306 / FAX: (304) 284-9001 World Wide Web: http://www.asset.com / e-mail: sei@asset.com

Copies of this document are available through the National Technical Information Service (NTIS). For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161. Phone: (703) 487-4600.

This document is also available through the Defense Technical Information Center (DTIC). DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center / Attn: BRR / 8725 John J. Kingman Road / Suite 0944 / Ft. Belvoir, VA 22060-6218 / Phone: (703) 767-8274 or toll-free in the U.S.: 1-800 225-3842.

Table of Contents

List of Figures

List of Tables

Abstract

Why are beneficial software engineering practices not being used effectively in the development of software systems? This question has intrigued researchers in software engineering for many years [Parnas 85]. Billions of dollars per year are spent, and a large proportion wasted, on building and maintaining software systems that are either never completed or, if completed, are of poor quality. This state of software development has led to the introduction of innovative tools and techniques to support the software development process. Initial evidence from use of these tools and techniques shows significant improvements in development productivity and software quality. However, many of these potentially beneficial tools and techniques have not been widely adopted or diffused. This research seeks to examine the reason for *why* this is so: What factors explain the successful diffusion of new software development techniques into practice?

Software development techniques are viewed as a subset of the broader category of information technology (IT). A research framework is developed that explains the complex relationship between developer involvement in the IT adoption process, characteristics of the environment into which the IT is introduced, and IT diffusion success. The framework posits that the effects of developer involvement and IT diffusion environment characteristics on IT diffusion success are mediated by (1) the developer's perceived control over his work when using the IT, (2) the developer's perceptions of the IT, and (3) the developer's perceptions of the impacts of IT use. Using the Personal Software Process[SM] (PSP[SM]) approach as an example of an innovative IT, we develop a survey instrument for the collection of study data, which are analyzed using path analysis. The survey was distributed to software developers who have used PSP on software development projects in industry. Results find support for mediating effects of Perceived Control on the relationships between Developer Involvement, Training, Voluntariness, Champion Support, and IT Diffusion Success. Results also find direct impacts of Developer Involvement, Training, and Voluntariness on IT Diffusion Success.

This research contributes to research by providing a model that integrates knowledge from the research fields of information systems (IS), diffusion of innovations, software engineering, and social psychology to better understand why user involvement and diffusion environment impact diffusion success. Further, it provides a measure for perceived control in an IS context. This research contributes to practice by stressing the importance of software developer perceptions in determining IT diffusion success. The research underscores the importance of creating a perception of greater control over the software development process in order to positively influence diffusion success. Guidance is provided to software develop-

[SM] Personal Software Process and PSP are service marks of Carnegie Mellon University.

ment practice by stressing the importance of software developer involvement, training, voluntariness, and champion support in the introduction of new IT. These variables can be influenced by management to enhance perceived control and, therefore, better ensure successful diffusion of new software development practices in organizations.

Editor's Note: This SEI Special Report also includes information from a presentation delivered by the authors at the Software Engineering Institute. The information, which appears in Appendix B, is material collected by the authors from their questionnaire and specifically concerns the Personal Software Process^{SM} *(PSP*^{SM}*).*

1 Introduction

Why are beneficial software engineering ideas not being used effectively in the development of software systems? This is a question that has intrigued and confounded researchers in the software engineering field for many years [Parnas 85]. It is widely accepted that current software development practices have led to a "software crisis" that threatens the future promise of the information age [Gibbs 94]. Billions of dollars are spent, and large proportions wasted, every year on software systems that are either never completed or, if completed, are of poor quality.

The primary motivation for this research is to improve efforts to introduce and diffuse new and effective software development techniques in software development organizations. While the software engineering field has been successful in developing new tools and techniques to improve the software development effort, software development continues to be less than optimal. Many of the new tools and techniques have been shown to improve software development, but are still not widely adopted and, when adopted, are often abandoned. Why this happens (and, thus, how this can be prevented) is the issue explored in this research.

The authors have personally observed situations in which seemingly beneficial software development tools and techniques were introduced to development organizations, yet their use in the organization did not "take off" as expected. These personal examples include the use of metrics, upper-CASE tools, and code/object reuse. The importance of this issue has been noted in IS and software engineering research on CASE tools [Orlikowski 93, Chau 96, Iivari 96], object-oriented software development techniques [Fayad 96], user-centered development techniques [Mantei 89], and formal development techniques in general [Holloway 96]. In the instance of CASE tool usage, Iivari [Iivari 96] noted that while CASE tools are "claimed to increase ... productivity of systems development and the quality of the developed systems ..., the actual use of CASE technology has been much less than one would expect" (page 94). He notes that a year after introduction, 70% of CASE tools are never used by individuals in the IS organization, and only 5% are widely used, although not to the full capability of the tool. Similar assessments exist for other software development techniques that have been demonstrated in research and practice to improve software development effectiveness, but have not been put into widespread use.

The motivation for this research, therefore, is to better understand the phenomena described above, and use that understanding to better inform software development researchers and practitioners on how to ensure the successful use and diffusion of beneficial software development techniques in their IS organizations.

Thought leaders in the software development community, including Constantine, DeMarco and Lister, and Yourdon, recognize that solutions to the problems of software development must include the human dimension (i.e., peopleware and good management practices) as well as the technical dimension [Constantine 95, DeMarco 87, Yourdon 96]. Thus, a basic premise of this research is that an effective strategy for information technology (IT) transition must integrate the technical research in software engineering with the relevant behavioral research in information systems (IS).

A large body of research on the transition and use of IT in organizations has been produced by IS researchers. The concept of IT is expanded in this research to include software development techniques, which can be innovations to the community of software developers. Thus, we can study software development practices under the guidance of well-established IS theory. A contribution of this research is to integrate behavioral IS research on IT transition and use with software engineering research on IT transition. This is accomplished through the development of a comprehensive framework for Software Development Technique Diffusion Success. This framework (and the research study based on it) shows that two key factors influence software development technique diffusion success: (1) developer involvement in the implementation process, and (2) characteristics of the environment into which the techniques are introduced. Further, it is shown that the importance of these two factors is due to their influences on software developers' perceptions of control. It is the intent of this framework to provide a richer explanation of the diffusion of software development techniques in IS organizations.

1.1 Software Engineering Research on IT Transition

Around 1980, the software engineering field began to create a new subfield, technology transfer, to address issues surrounding the transition of technology from research and development (R&D) into broad use within organizations [Fowler 94]. Initially, the focus was on issues surrounding the transfer of technology from R&D into products. These issues focused on legal and procedural issues such as intellectual property, trademarks, licensing, and standards issues. However, recent attention has been given to more "social" issues in technology transfer from R&D into development. The *Communications of the ACM* (CACM) devoted a special issue to *Technology Transfer* (September 1996) in an effort to bring "human" and "organizational" issues to the forefront. This issue of CACM documented the ideas of several expert practitioners in technology transfer. A list of success factors in technology transfer was presented. However, the focus of technology transfer in this special issue is from R&D into product development.

The Software Engineering Institute (SEI) supports an ongoing project to study a broader scope of IT transition issues. Fowler and Levine [Fowler 93a] as shown in Figure 1, propose a framework and model of software technology transition. In this model, technology transition is said to consist of three overlapping cycles: research and development (R&D), new

product development, and adoption/implementation. Each of these cycles consists of various activities as presented in Table 1. The two intersections of the three cycles represent transactions in the model. The *transfer* of a technology from R&D to product development includes developing the standards and skills to turn the initial theoretical concept into a useful, viable product. The diffusion of the technology product is the process of prescribing how people adopt and implement the technology over time into their organizations. The focus of this research is to better understand the technical and behavioral issues of *diffusing* innovative software development techniques into practice. The research draws on literature in information systems, software engineering, and social psychology in order to develop a research framework for the Diffusion of Software Development Techniques in IS Organizations.

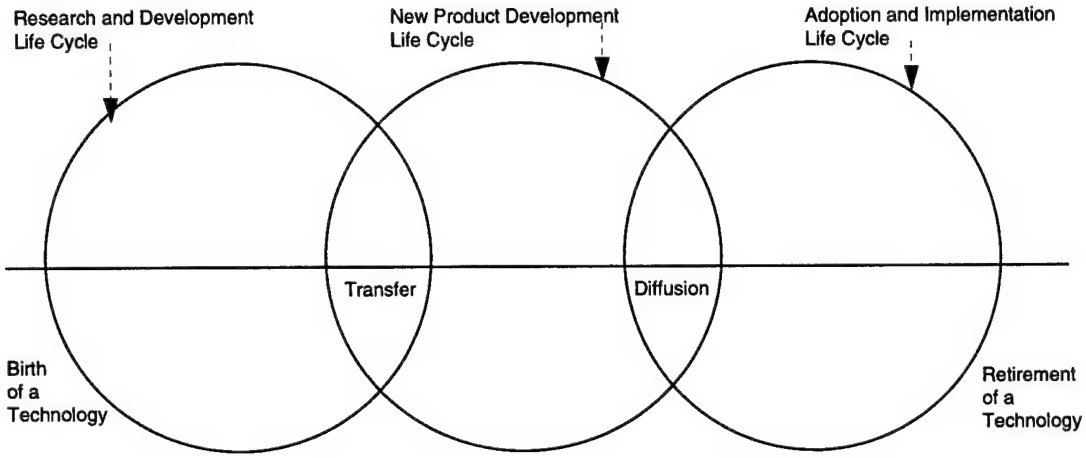


Figure 1: Framework for Technology Transition

[Fowler 93a]

Research and Development Activities: • Concept Formulation • Development and Extension • Enhancement and Exploration • Early Popularization
New Product Development Activities: • Generating New Product Ideas • Screening the Ideas • Testing Product Concepts • Business Planning • Development and Prototyping • Test Marketing and Pricing Strategies • Product Launch
Adoption and Implementation Activities: • Needs Assessment • Selection of Candidate Products • Evaluation of Candidate Products • Introduction of Selected Product to Management and End Users • Gathering of Feedback for Management and End Users • Implementation Planning • Implementation • Product Maintenance • End User Support

Table 1: Technology Transition Activities

[Fowler 93a]

1.2 Software Development Techniques as IT

Within the realm of technology transfer, the traditional scope of IT includes hardware, system software, and telecommunications components. This scope has been expanded in the software engineering community to include information systems, products, and technologies [Fowler 94]. The present research, however, expands this scope further by considering software development processes, techniques, and methods as ITs themselves. If we consider technology as tools that enable us to transform parts of our environment and extend our human capabilities [Tornatzky 90] then software development techniques can be considered tools that enable humans to transform ideas into solutions.

1.3 Software Development Techniques as Innovations

An innovation is an idea, practice, or object that is perceived as new by the individual or organization adopting it [Rogers 95]. The present research investigates the diffusion of recent, non-traditional software development techniques in organizations; thus, it studies software development techniques that are innovations in organizations. This specification of "recent," as opposed to a definitive timeframe, is purposely used because what may be considered new to some organizations may be considered established and traditional in other organizations. What matters is how the IT is *perceived* by the organization. Thus, the model set forth in this research could be used, for example, by an organization implementing software inspections for the first time, although such a technique may be an established practice at another organization.

1.4 Research Framework

From the standpoint of IT diffusion, a key dimension of successful diffusion of IT into practice is *use* of the IT by the organization [Rogers 82, Fowler 93a]. How then can we ensure an IT (i.e., a software development technique) will be used by a development organization into which it is introduced? What factors increase the successful adoption and use of a software development technique? These questions have been examined extensively in the IS and diffusion of innovations literature [Rogers 83, Davis 89a, Davis 89b, Mathieson 91, Moore 91, Adams 92, Fowler 93b, Hartwick 94, Taylor 95, Rogers 95]. In early work, Rogers identified five perceived characteristics of an innovation that affect the rate of diffusion of the innovation [Rogers 83]:

1. Relative Advantage
2. Compatibility
3. Complexity
4. Observability
5. Trialability

IS research has examined the impact of some perceived IT characteristics on IT use. Davis studied the impact of two of these characteristics, Perceived Ease of Use and Perceived Usefulness, on IT Use [Davis 89]. Perceived Ease of Use can be equated to Rogers' Perceived Complexity variable while Perceived Usefulness can be equated to Rogers' Perceived Relative Advantage variable [Moore 91]. Under the guidance of Davis' Technology Acceptance Model (TAM), Davis et al. found these variables to account for approximately 47% of the variance in IT usage. Taylor, in comparing the TAM with two other models predicting IT usage, found the TAM Perceived IT Characteristics variables to account for only 34% of variation in IT use [Taylor 95]. This finding led Taylor and Todd to suggest a broader exploration of factors that explain IT use, specifically suggesting the User Involvement construct as one that appears to be relevant to examining the use of IT innovations.

More recent work in IT diffusion from the software engineering community has identified factors in addition to the characteristics of the IT that impact diffusion success. In their case study on technology transition, Fowler and Levine [Fowler 93b] found five factors that impact diffusion success:

1. Role of the Change Agent and Key Players
2. Organizational Infrastructure
3. An Innovative Organizational Culture
4. Attributes of the Technology
5. Development of a "Whole Product"

While these factors give prescriptive guidance into diffusion success, this guidance is not linked into a descriptive model that explains *how* these factors impact diffusion success.

To summarize, five key points can be made: (1) an indicator of successful IT diffusion is IT use, (2) perceived attributes of an IT impact IT diffusion success, (3) organizational environment variables impact diffusion success, (4) variables, such as user involvement, are needed to augment perceived IT characteristics in order to better explain IT use, and (5) a descriptive model showing the relationships of these factors to IT diffusion success is needed.

The research framework in Figure 2 is proposed to study the diffusion of software development techniques. The research model that will be used in this dissertation research will be a subset of this framework. Both the framework and the research model variables will be discussed in more detail in Chapter 2.

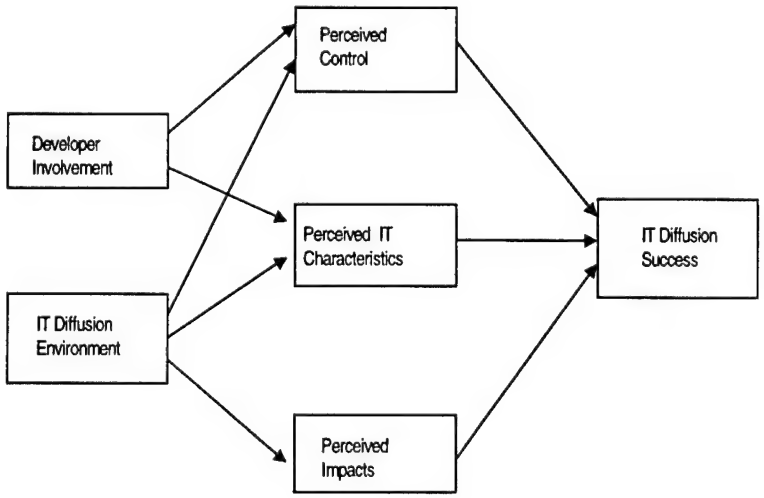


Figure 2: Diffusion of Software Development Techniques—Research Framework

1.5 Contributions of the Study to Research

The purpose of this study is to better understand the technical and behavioral issues that are important in diffusing innovative software development techniques into practice. In so doing, several important contributions to the research and practitioner communities are made. For the research community, this research provides a measure for software developer satisfaction. As will be discussed in later chapters, increased feelings of satisfaction have been linked with increased IT usage [DeLone 92, Baroudi 86]. Thus, in order to improve the use of effective software development techniques and, therefore, the effectiveness of software development, it is important to assess the satisfaction of developers. The IS literature abounds with measures of end-user satisfaction with the IS function and IS deliverables [Bailey 83, Ives 83, Doll 88]; however, there have been no previous measures targeted specifically to software developers as users of IT. This research fills this important gap in measurement.

This research also contributes to the research community by providing a measure of perceived personal control over work in an IS context. Again, there have been no previously developed measures of perceived personal control in an IS context, although IS research has called for measures in this area [Baronas 88]. The measure of perceived control combines the concepts of perceived control from social psychology as well as management literature to form a more robust measure of perceived control.

Finally, to the research community, this research provides an integrated model for IT diffusion success. By combining key concepts from the Software Engineering, IS Implementation, Diffusion of Innovations, and Social Psychology literatures, the descriptive research framework developed in this study increases our understanding of the diffusion of IT in organizations. Specifically:

- Software Engineering literature is used to identify IT innovations that can improve software development.
- IS Implementation literature is used to identify key indicators of successful IT diffusion.;
- Diffusion of Innovations and IS literature are used to identify behavioral and organizational issues that impact successful diffusion of IT.
- Social Psychology literature is used to explain why the behavioral and organizational issues impact successful IT diffusion.

1.6 Contributions of the Study to Industrial Practice

This study contributes to industrial practice in at least two important ways. First, the importance of developers' perception of control over their work when using innovative software development techniques is explored. Support for the research hypotheses points to a need for IS management to include steps to ensure that software developers maintain a sense of control over their work when introducing new software development techniques. Second, the

research framework provides insight on increasing this perception through the increased involvement of software developers in the IT diffusion process, and through specific aspects of the organizational environment. Thus, the overall contribution to IS practice is in providing guidance to increase the success of innovative software development techniques in IS organizations. This guidance can be used effectively by organizational change agents, software development managers, and software developers themselves.

1.7 Reader Roadmap

The organization of this report is as follows. Chapter 2 reviews the literature, which serves as the basis for the development of the research model. From the review of this literature, research propositions are developed. Chapter 3 discusses the research methodology that is used to study the research propositions presented in Chapter 2. This discussion includes the operationalizations of the constructs defined in Chapter 2, as well as evaluations of the psychometric properties of the study survey instrument. Chapter 4 discusses the results of analysis of the study data. Finally, Chapter 5 gives concluding remarks on this study, including contributions and limitations of this research, and opportunities for future research.

Researchers in the area of technology transition will want to read the complete report with special emphasis on Chapters 3 and 4 where the research experiment is designed and the resulting data analyzed. Practitioners who desire guidance on how best to introduce a new IT into their organization will want to focus on the conclusions provided in Chapter 5. Those who want to understand the issues of technology transition and the background literature should study Chapter 2 where we provide a comprehensive literature survey and present a comprehensive research model of technology transition. Finally, there is much research yet to do in this area. Researchers who are interested in extending this work should read Chapters 2 and 5 where the research model is presented and future research directions are proposed.

2 Literature Review

2.1 Research Literature Base

This chapter reviews the literature that serves as background for the constructs and relationships depicted in the research framework. This research draws from a number of sources:

1. The definition of IT Diffusion Success is guided by *IS Implementation* literature on IT Success. This literature is rich with theory-backed conceptualizations of IT Success as well as validated measures of IT Success. The constructs taken from this literature include IT Use and Satisfaction with the IT.
2. The key factors that impact IT Diffusion Success are drawn from literature in *Information Systems and Diffusion of Innovations*. These factors include User Involvement, and Characteristics of the Environment into which the IT is introduced.
3. To aid in understanding *why* the key factors identified above impact IT Diffusion Success, we draw from literature in *Social Psychology, IS Implementation*, and *Software Engineering*. The mediating constructs that have been identified include Perceived Control over Work, Perceived IT Characteristics, and Perceived Impacts from IT Use.

The remaining sub-sections of this chapter review the literature referenced above, and develop research propositions used to guide the dissertation research. Figure 3 shows the research variables and propositions that will be described in this chapter.

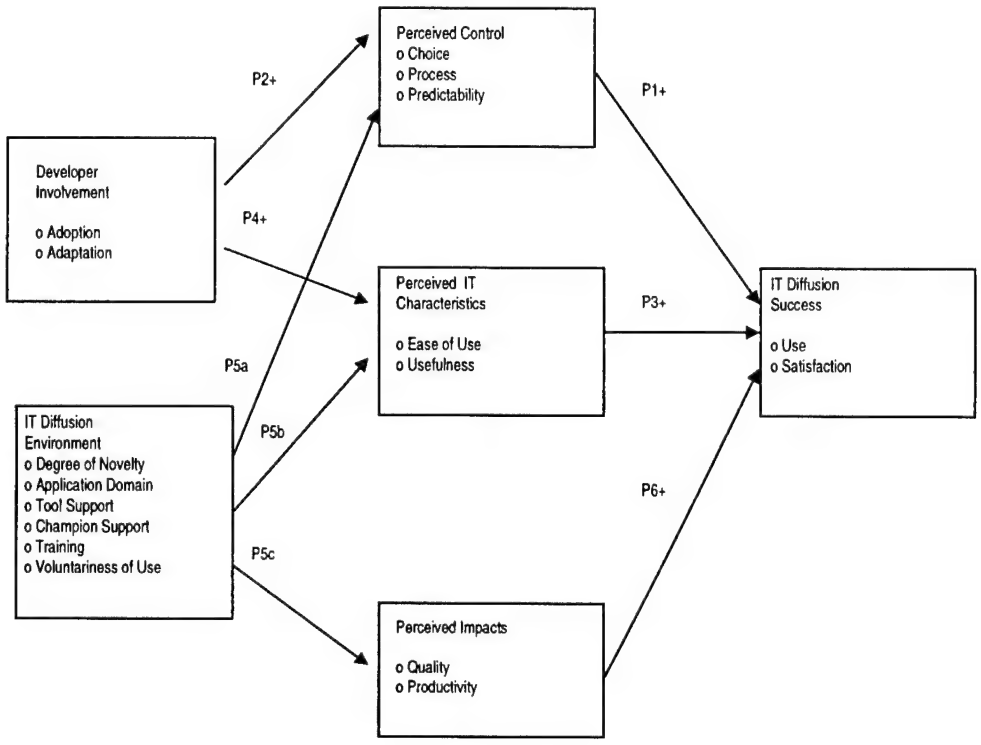


Figure 3: Research Framework—Variables and Propositions

2.2 IT Diffusion Success

A key measure of the successful diffusion of an IT in an organization is the use of the IT [Rogers 82, Fowler 93a]. In their review of IS research on IS Success, DeLone and McLean observe that IT use is the most frequently reported measure of IT implementation success [DeLone 92]. Thus, IT Use[1] has been chosen as one of the indicators of IT Diffusion Success in the proposed research framework.

When use of an IT is mandated, as is often the case when introducing new software development technologies, Satisfaction has been suggested as a more appropriate measure of IT Success [DeLone 92, Adams 92, Melone 90, Chau 96]. Chau observed that in many software development innovation adoption scenarios, the adoption decision is made at an organizational level and software developers are usually required to use it [Chau 96]. However, he notes that the issue of whether or not the software developer enjoys using the innovation, (i.e., his satisfaction with the innovation) is of critical importance to productivity. Thus, Satisfaction with the IT has been chosen as a second IT Diffusion Success variable in the framework.

[1] Throughout this paper, we generally begin research framework variables with capital letters.

2.3 User Involvement

End-user involvement has long been endorsed by the information systems research community as a way to ensure successful IT implementation [Powers 73, DeBrabander 77, Ives 84, Baroudi 86, Doll 89, Hartwick 94]. User involvement has been defined as participation in the systems development and implementation processes by representatives of the target user group [Ives 84]. Involvement by users is generally deemed most critical during requirements definition and logical design phases [Ives 84, Doll 89]. User involvement is most commonly assumed to increase user satisfaction and IT Use [Baroudi 86]. Ives and Olson develop a descriptive model of user involvement based in part on the work of Locke and Schweiger [Ives 84, Locke 79]. Grounded in theory on participative decision-making and planned organizational change, the model defines six degrees of user involvement corresponding to the amount of influence the user has over the final product. The degrees of involvement range from no user involvement in the development process to the user as a design team member. The Ives and Olson model also includes two psychological factors, cognitive and motivational, which explain the effects of user involvement on user satisfaction and IT use. Cognitive factors address increases in knowledge and understanding that user involvement can bring to users. Motivational factors address greater acceptance of and commitment to decisions and changes by users due to such things as greater feelings of control. Thus, users' perception of control is noted as an important variable to explain the effects of user involvement on IT use. However, as noted by Ives and Olson, little user involvement research has made the linkages between user involvement, the psychological factors, and satisfaction/IT use explicit [Ives 84]. Two notable exceptions to this are the Doll and Torkzadeh, and Hartwick and Barki studies [Doll 89, Barki 94].

Doll and Torkzadeh present a model of Psychological Mechanisms that link end-user involvement to end-user satisfaction [Doll 89]. In their model, they distinguish between two types of user involvement—desired involvement and perceived involvement—and posit that the interacting effects of these two variables are mediated by cognitive, motivational, and value attainment psychological variables. However, as with the Ives and Olson model, the Doll and Torkzadeh model is presented but not subsequently tested [Ives 84, Doll 89].

Hartwick and Barki used the Theory of Reasoned Action (TRA) model to examine the relationship between user involvement (referred to as user participation in their study) and IT use [Hartwick 94]. The TRA is a theory from social psychology, which states that one's behavior (i.e. Use) is a function of one's intentions, and that intentions are determined by one's attitude and the subjective norms concerning the behavior [Fishbein 75]. Hartwick and Barki attempted to explain the effect of user involvement on IT use by dividing the traditional notion of user involvement into two constructs—user participation and user involvement. User Participation refers to "the behaviors, assignments, and activities that users or their representatives perform during the [Information Systems Development] process" [Hartwick 94, page 441]. In other words, user participation represents the traditional notion of user involvement, which is behavior-oriented. On the other hand, user involvement, as defined by Hartwick and Barki, refers to "the extent to which a person believes that a system possesses...importance

and personal relevance" [Hartwick 94, page 442]. In other words, user involvement represents an attitude or psychological state. Although methodological limitations of the study prohibit strong conclusions, their empirical study did find a significant relationship between user involvement and IT use. Moreover, they found support for the psychological construct of user involvement as a mediating variable between user participation and IT use.

In summary, IS research on user involvement informs the present research through (1) the identification of user involvement as a key determinant of IT use and satisfaction, and (2) the understanding of how user involvement impacts IT use and IT satisfaction via psychological constructs, such as users' perception of control.

2.3.1 Developer Involvement

Previous work in user involvement focuses on end-users of IT as the population of interest. This study, however, will focus on software developers as the population of interest. Although the definition of user involvement by Ives and Olson includes participation in the systems development and implementation processes, the context of most empirical work in user involvement research has focused on end-user involvement in the development of the IT [Ives 84, Baroudi 86, Tait 88, Doll 89, Hartwick 94]. As noted earlier, the present research focuses on the diffusion of software development techniques. Consistent with research on technology transition, when studying diffusion of an IT it is assumed that the IT has previously been designed and developed by R&D and product development groups. In this vein, we assume that the software development techniques being investigated will have been previously designed and developed by R&D groups in university or industry settings. Under this assumption therefore, the notion of software developer involvement in the development of software development techniques becomes non-applicable. However, if we adopt the context of "participation in ...the [IT] implementation process" as stated in the Ives and Olson definition, then we can study software developer involvement in the implementation of software development techniques [Ives 84].

Kwon and Zmud divide the IT implementation process into six steps [Kwon 87]:

1. *Initiation*, the pressure to change, leading to gathering and evaluation of information
2. *Adoption*, the decision to commit resources to the IT innovation
3. *Adaptation*, the development and installation of the IT innovation
4. *Acceptance*, the user's attitude towards the use of the IT innovation
5. *Use/Satisfaction/Performance*, IT innovation implementation success variables
6. *Incorporation*, when the IT is embedded within the organization's routine

The research framework focuses on software developer involvement in the initiation, adoption, and adaptation of the software development technique in the developer's organization.

2.4 Developer's Perception of Control

Research on user involvement has suggested that psychological mechanisms, such as greater perceptions of control, help explain the impact of user involvement on IT success. However, little research in user involvement has studied this linkage. A few IS studies have explored perceived control in an IS context, using the Theory of Planned Behavior (TPB). Mathieson and Taylor and Todd examine the TPB in predicting IT Use [Mathieson 91, Taylor 95]. TPB includes Social Norms and Perceived Behavioral Control as factors that influence Behavioral Intention to Use the IT (see Figure 4) [Ajzen 85, Ajzen 91]. In their study of usage of a computing resource facility, Taylor and Todd found that Perceived Behavioral Control was a significant determinant of both Behavioral Intention to Use the IT and Actual IT Use [Taylor 92]. Taylor and Todd as well as Mathieson conclude that TPB, with its inclusion of Social Norms and Perceived Behavioral Control variables, should be the preferred model for better *understanding* Usage behavior and guiding development and implementation decisions for new ITs.

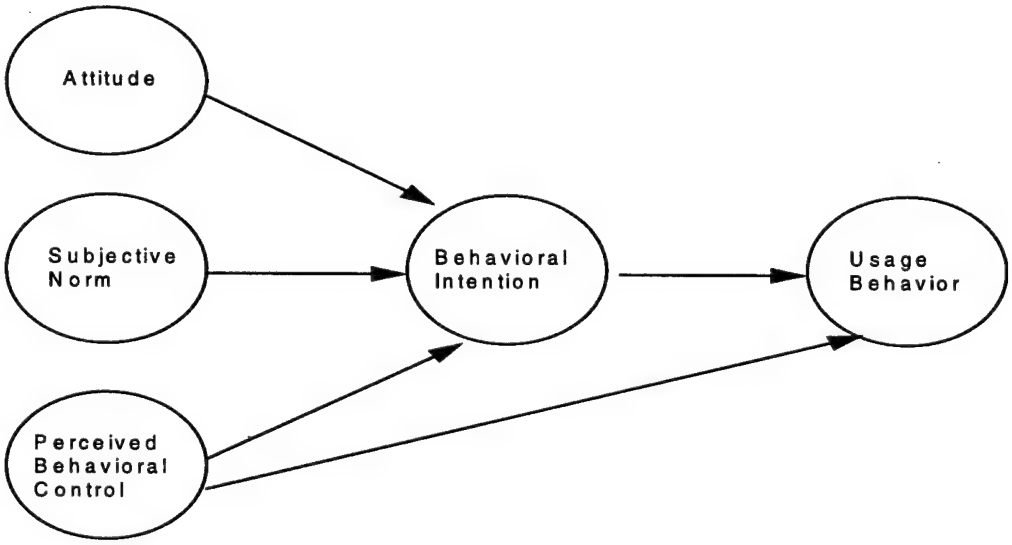


Figure 4: The Theory of Planned Behavior

[Ajzen 85, Ajzen 91]

While the TPB model is significant in establishing a user's perception of control as a key construct in understanding whether or not an IT will be successfully implemented in an organization, the model as-is does not give guidance to those wishing to influence a user's perception of control. The model takes a limited view of a user's perception of control, focusing on behavioral control only. The present research focuses on two additional dimensions of perceived control: perceived choice control and perceived predictability.

Further, the operationalization of perceived behavioral control in an IS context has traditionally focused on capturing beliefs regarding access to resources and opportunities needed to perform a behavior, and/or internal factors that may impede performance of the behavior

[Taylor 95, Mathieson 91]. This focus has ignored external factors that may be perceived to impede performance of the behavior. These external factors could take the form of managerial behavioral control where management influences developers to use an IT in a certain way, and/or behavioral constraints placed on a developer that may be a result of the design of an IT. The present research framework takes this broader view of perceived control from the social psychology literature. This approach provides us with a richer view of perceived control that will enable us to better understand the construct and thus better identify potential factors that can be expected to influence it.

The construct Perceived Control comes from social psychology research. An individual is said to seek control because of the need to know causes and consequences of his and others' behavior [Baronas 88]. Perceived control is usually defined in terms of three dimensions: (1) decisional or choice, (2) process or behavioral, and (3) predictability [Langer 83, Baronas 88]. The decisional or choice dimension refers to an individual having the opportunity to choose among various possible actions. Early research on control equated more choice with more control; subsequent research, however, found that too many choices can overwhelm and hence reduce perceived control [Langer 83]. Still, increased control via choice has been found to enhance performance on learning tasks and to reduce stress [Monty 73, Perlmuter 71, Corah 70].

The process or behavioral view of control refers to the ability of an individual to take direct action on the environment to influence an event. It represents the ability to, for example, change or escape an unpleasant situation, or to determine the sequencing of events. This ability, however, could be constrained by external factors such as availability of resources, managerial constraints, constraints imposed by the IT, and the like. An individual's perception of his own process control has been linked to such outcomes as enhanced perceptions of trial justice [Giacobbe-Miller 95]. Other studies have found that greater managerial behavior control results in higher performance on IS development teams [Henderson 92]. In their survey of 79 software designers representing 41 development teams, Henderson and Lee found that those teams who noted greater levels of managerial behavioral control had higher levels of performance [Henderson 92]. Neither the Henderson and Lee study nor other studies reviewed by the author on the subject of perceived behavioral control examined *perceived behavioral control imposed by the IT* being used [Henderson 92]. The present research examines perceived process control as the degree to which a software developer feels that his ability to perform software development tasks when using the IT is free from constraints.

The predictability dimension of control refers to knowing what event(s) will occur and when, and not necessarily to controlling the event itself. Enhancing an individual's level of predictability has been found to decrease stress and increase satisfaction [Langer 83].

The above research suggests the following proposition:

Proposition 1: A positive, significant relationship exists between software developers' Perceived Control over their work when using an IT innovation, and the IT Diffusion Success.

Baronas and Louis studied issues of perceived control in an IS context [Baronas 88]. In particular, they investigated perceived control as a mechanism through which user involvement in the IS development process impacts the user's satisfaction with IT. An exploratory field experiment was designed such that the standard development process was changed to increase treatment subjects' personal control via manipulation of choice and predictability. Baronas and Louis suggest that enhanced perceptions of Choice are associated with increased levels of Satisfaction, although this link is not directly examined [Baronas 88]. This research examines this link with the following hypothesis:

H1: As Software Developers' perceptions of choice in using a new IT increase, their level of Satisfaction with the new IT will increase.

No previous studies have been found that directly test the relationship between Perceived Choice and Use. However, Baroudi et al. found that greater levels of Satisfaction with an IT lead to greater levels of IT Use [Baroudi 86]. Thus:

H2: As Software Developers' perceptions of choice in using a new IT increase, their level of Use of the new IT will increase.

Tetrick and LaRocco studied the effects of predictability and process control on job satisfaction [Tetrick 87]. They found that when workers felt they had greater influence over the events in their work environment, they were more satisfied in their jobs. Thus, it is expected that if software developers experience greater degrees of process control, then their Satisfaction with a new IT will increase; i.e.:

H3: As Software Developers' perceptions of process control in using a new IT increase, their level of Satisfaction with the new IT will increase.

Taylor and Todd found a significant relationship between Perceived Behavioral Control (or process control) and IT Use [Taylor 95]. However, as noted previously, their operationalization of Perceived Behavioral Control did not account for external constraints on behavior such as managerial behavioral control or constraints on behavior by the IT itself. This research operationalizes process control as freedom from constraints when using a new IT to perform software development tasks. Consistent with Taylor and Todd and Mathieson, the following is hypothesized [Taylor 95, Mathieson 91]:

H4: As Software Developers' perceptions of process control in using a new IT increase, their level of Use of the new IT will increase.

Regarding the relationship between Predictability and IT Success, Tetrick and LaRocco posited a direct association between predictability and job satisfaction, based on previous research in work stress health [Tetrick 87]. Although positive, moderate correlations are found between the two, confirmatory factor analysis suggests no direct relationship. However, somewhat low reliability of their Predictability scale (Cronbach's alpha = .66) suggests that subjects' true assessment of predictability may not have been adequately measured. Thus, this research continues to examine this direct association between perceived Predictability and Satisfaction in the following hypothesis:

H5: As Software Developers' perceptions of predictability using a new IT increase, their level of Satisfaction with the new IT will increase.

No empirical studies have been found that directly examine the relationship between Predictability of events and Use. However, Fowler and Levine suggest this relationship in their case study of the diffusion of a new software development IT, Rate Monotonic Analysis scheduling (RMA) [Fowler 94]. Fowler and Levine present comments from developers of the adopting organization regarding their experiences with initial use of RMA. One reason given for continued use of the IT was their confidence in "always know[ing] how it will work." In other words, predictability in their work environment contributed to subsequent use. Therefore, it is hypothesized that greater perceptions of predictability in software development tasks when using a new IT will be associated with greater levels of Use of the IT:

H6: As Software Developers' perceptions of predictability using a new IT increase, their level of Use of the new IT will increase.

Baronas and Louis posited in their study of perceived control that users more involved in the software development process would report greater feelings of perceived control [Baronas 88]. Again, Baronas and Louis did not directly examine this link; this research will examine this relationship. The following proposition is offered:

Proposition 2: A positive, significant relationship exists between software Developer Involvement in the initiation, adoption, and adaptation of an IT innovation, and the software developers' Perceived Control over their work when using an IT innovation.

The initiation and adoption phases of the implementation process result in a decision to adopt or not adopt an IT innovation. The more involved a software developer is in this decision process, the more likely he is to perceive greater levels of choice in whether or not to adopt the IT. Thus, we can expect greater levels of involvement in the adoption process to result in greater levels of perceived choice in adoption. The hypothesis to be tested is as follows:

H7: As Software Developers are more involved in the new IT adoption process, their perceived choice in using the new IT will increase.

2.5 Perceived IT Characteristics

As noted in Chapter 1, traditional innovations and IS research have focused on the role of Perceived IT Characteristics in explaining IT adoption and use. Especially prevalent are studies that examine how perceptions of the IT as well as attitudes and beliefs about IT use can impact IT usage. Davis, Mathieson, Adams, and others have demonstrated empirical support for the Technology Acceptance Model (TAM) [Davis 89a, Davis 89b, Mathieson 91, Adams 92]. TAM is an adaptation of the Theory of Reasoned Action (TRA), which states that two beliefs, Perceived Usefulness of the IT and Perceived Ease of Use of the IT, impact Attitude toward IT Use [Fishbein 75]. TAM further states that Attitude toward IT Use impacts a user's Behavioral Intention to Use the IT which, in turn, predicts use of the IT. Figure 5 shows a depiction of the TAM model. The two Perceived IT Characteristics used in the TAM model have been shown to account for approximately 34%–47% of variance in IT Use behavior [Taylor 92, Davis 89b]. These findings support diffusion theory that states that Perceived IT Characteristics impact an IT innovation's rate of adoption. Thus:

Proposition 3: A significant, positive relationship exists between software developers' Perceived IT Characteristics and the Diffusion Success of the IT.

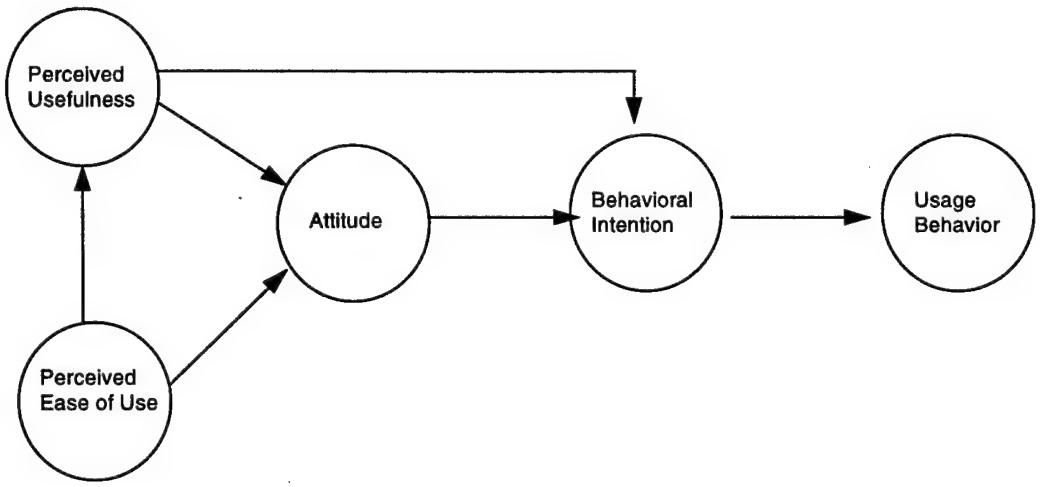


Figure 5: Technology Acceptance Model

[Davis 89]

Taylor and Todd found that a significant percentage of variance in IT Use remained unaccounted for by Perceived IT Characteristics alone [Taylor 95]. They suggested an examination of the User Involvement construct as another possible explanation for IT Use. Recently, Jackson et al. studied the impacts of User Involvement (which they referred to as situational involvement) on Perceived Usefulness hypothesizing a positive relationship between these two constructs [Jackson 97]. Although their analysis found a positive relationship between

the constructs, the effect was not significant. However, their analysis was conducted using the LISREL technique with a sample size of only 111 observations; thus, the power of their study to detect a significant effect may have been limited using this statistical technique. Thus, this relationship should be studied further:

Proposition 4: A significant, positive relationship exists between software Developer Involvement and Perceived IT Characteristics.

2.6 IT Diffusion Environment

Research in IS and Diffusion of Innovations has suggested that Organizational and Environmental characteristics impact the actions, reactions, and perceptions of users [Davis 89, Orlikowski 93, Rogers 95, Iivari 96]. Davis refers to these characteristics as "externally controllable factors" and includes such things as development methodologies, training and education, as well as the aforementioned user involvement [Davis 89a]. Orlikowski, in her studies of CASE tool adoption and use, found that individual, organizational, and IT characteristics influence software developers' responses to CASE tool use [Orlikowski 93]. These responses ranged from behavioral (e.g., resisting the use of CASE tools altogether) to perceptive (e.g., belief about CASE strategy).

The research framework developed in this study includes Characteristics of the Environment into which the software development technique is being introduced. As the following literature review reveals, some of these environmental characteristics can have a positive impact on user reactions, while others can have a negative impact. Thus, the following general proposition is offered:

Proposition 5: A significant relationship exists between characteristics of the IT Diffusion Environment and software developers' perceptions.

This proposition can be broken down further by accounting for the different perceptions identified in the research framework:

Proposition 5a: A significant relationship exists between characteristics of the IT Diffusion Environment and software developers' Perceived Control over their work when using the IT.

Proposition 5b: A significant relationship exists between characteristics of the IT Diffusion Environment and software developers' Perceived IT Characteristics of the IT innovation.

The IT Diffusion Environment construct includes various individual, organizational, and environmental characteristics surrounding the use of the software development technique (IT). Numerous individual, organizational and environmental characteristics have been identified in previous research. However, this research framework identifies those characteristics that

are of particular importance to the diffusion of innovations: the Degree of Novelty of the Innovation, the Application Domain to which the innovation is being applied, the availability of Tool Support for the innovation, the degree of Champion Support for the innovation, the effectiveness of Training on use of the innovation, and the degree of Voluntariness of Use of the innovation. These variables are discussed in the following six sub-sections.

2.6.1 Degree of Novelty

Degree of Novelty refers to the extent to which the learning and use of the IT innovation represents a new experience to the user. Chau refers to this as the implementation gap, where the gap is the difference between current skills and knowledge, and the skills and knowledge required by the new IT [Chau 96]. In a field survey conducted by Chau, he found a significant, negative relationship between the degree of novelty and IT acceptance [Chau 96]. Further, he found that this negative influence was through its effect on two perceived IT characteristics: Perceived Ease of Use and Perceived Usefulness. Thus, Chau has demonstrated that greater degrees of novelty of an innovation have a negative impact on perceived IT characteristics and, therefore, on IT acceptance.

Similarly, Orlikowski identified a variable, Nature of Change, which was found to impact software developer's perceptions of CASE tool use [Orlikowski 93]. Nature of Change can be incremental (an extension of the status quo, consistent with current practices, skills, and norms), or radical (a shift to fundamentally different practices, skills, and norms). Orlikowski noted that radical change is often associated with ambiguity and chaos. Put in the context of the adoption of software development techniques, a technique (or suite of techniques) that represents a radical change to the software developer can lead to a work environment that is uncertain, unclear, and thus less predictable than more familiar techniques. This reasoning leads to the following hypothesis:

H12: As Software Developers' Degree of Novelty of the new IT increases, their perceived predictability with the new IT will decrease.

2.6.2 Application Domain

The Application Domain of the IT represents characteristics of the problem being addressed by the IT. In the context of software development techniques, it represents the criticality and definition of the development effort to which the technique is being applied. The software engineering literature on technology transfer has suggested that some software development tools and practices may be beneficial only within the context of certain application domains [Holloway 96, Luqi 97]. In their discussion of impediments to industrial use of formal development methods, Holloway and Butler note that application domains that are used to develop and test the viability of new software development tools and practices are often irrelevant to industry [Holloway 96]. The implication is that as long as the application domain in practice is consistent with the application domain used in research and development of the IT, that

beneficial use can result. Holloway and Butler, however, call for more support from researchers for models that can be used in those application domains that are critical to IS practice.

Luqi and Goguen also study the applicability of formal software development tools and techniques to application domains [Luqi 97]. They echo Holloway and Butler's call for more practical, less trivial models to guide technique use in practice. However, they further emphasize the use of these tools and techniques in "narrow, well-defined, and well-understood problem domains" (page 81). Thus, critical, well-defined application domains have been suggested as positive influences on acceptance of IS development practices. This research adopts Holloway and Butler's and Luqi and Goguen's specification of application domain by focusing on the extent to which the software development innovation is used for critical, well-defined applications.

2.6.3 Tool Support

The availability of Tool Support is another environmental characteristic of importance to the diffusion of software development techniques. Holloway and Butler suggest that the unavailability and inadequacy of tool support represent serious impediments to widespread use of formal development methods [Holloway 96]. Fayad et al., note that the use of tools to support software development techniques enables software developers to focus on the analysis and design tasks of software development, as opposed to "trivial" tasks such as transposing information from one place to another, conforming with syntax rules, producing graphics, reports, etc. [Fayad 96] They report that lack of tool support may result in insecurities about the success of the new software development process. The present research will examine the availability of tool support for the software development innovation.

2.6.4 Champion Support

The importance of Champion Support for IT diffusion efforts has been noted in IS research [see Prescott 95]. An IT Champion has been defined as a "manager who actively and vigorously promotes their personal vision for using IT, pushing [a] project over or around approval and implementation hurdles" [Beath 91]. Beath speaks of IT champions as *transformational leaders* who are distinguished from "ordinary" managers or IT sponsors in that the latter use access to funds and authority to accomplish a goal, whereas the former use a variety of other influences (social, political, etc.) to accomplish goals.

Most empirical studies on IT diffusion and use do not approach the issue of Champion Support directly. However, a great number of IS and software engineering studies acknowledge the importance of Management Support of the IT innovation [Hoffer 92, Wynekoop 92,Rai 94, Fowler 95, Fayad 96, Chau 96, Iivari 96]. In his study on CASE tool use, Iivari identifies Management Support as a critical variable influencing the diffusion of CASE tools [Iivari 96]. Management Support is defined as consisting of two aspects: (1) controlling resources needed for system development and use, and (2) the signaling of clues to encourage certain behaviors. In his field study, Iivari found Management Support to be significantly and posi-

tively associated with CASE usage. In Iivari's study as well as most prior studies in this area, the level of analysis has been at the organizational level.

One empirical study that does approach the issue of Champion Support directly is a study conducted by Chau [Chau 96]. Chau identifies two key aspects of Management Support as (1) providing training for users, and (2) reducing or eliminating barriers to use. In a survey of 97 software developers, Chau found Management Support (as a part of what he referred to as "transitional support") to be positively related to Perceived Ease of Use. Chau further recommended more Management Support in initial stages of IT adoption.

The above characterizations of Management Support closely resemble the definition Beath gives for Champion Support. Indeed, Alexander, in her research on the adoption of database machines, found Management Support and Champion Support to be highly correlated [Beathe 91, Alexander 89]. The difference between IS Management Support and Champion Support, as noted by Beath, is in the *way* the IT Champion accomplishes his goals. However, this research will be concerned with the *existence* of a champion for the IT innovation, not the way in which the champion accomplishes his goals.

Alexander describes champions as "individuals who advocate the adoption of an innovation," and have an ability to influence decisions [Alexander 89]. The degree to which a champion is able to influence decisions can be expected to influence the degree to which a developer feels free to make his own decision (choice) on adoption. Thus:

H8: As Software Developers' perceptions of degree of Champion Support for a new IT increase, their perceived choice whether or not to use the new IT will decrease.

Support for the above hypothesis is not an argument against Champion Support. To the contrary, champion and management support have been deemed beneficial in increasing IT use [Iivari 96, Chau 96]. However, it may suggest the importance of balancing the potentially negative affect on Perceived Control with emphases on enhancing Perceived IT Characteristics and Impacts (other constructs depicted in the research framework), as well as incentives and other motivators (not included in the research framework) in order to ensure developer Satisfaction with and sustained use of software development techniques.

2.6.5 Training

The availability and adequacy of training represents a crucial factor in the successful diffusion of software development innovations [Fayad 96]. Training on the use of the IT is an essential prerequisite to putting the IT to use. However, a key consideration is the context within which training is conducted. Gersick and Hackman, in their research on habit-forming, note that the most effective way to introduce an intervention (i.e., a change in the way of doing things) is when the individual or group is at a "breakpoint" in work [Gersick 90]. They note that when groups are required to *change* their norms and routines when they are simultaneously occupied with *performing* the routine, the effect is distracting and unhelpful at best.

Based on numerous experiences with transitioning to object-oriented development, Fayad et al. similarly note the importance of training on new techniques that occurs while no active project work is required [Fayad 96]. They argue that full concentration should be given to acquiring the new "habits," and learning the new techniques.

This research will focus on the availability, adequacy, and context of training as described above. Note that there is much research in IS that examines the nature of training and its impact on various individual outcomes, such as performance [Santhanam 94, Davis 93]. As the present study focuses on Satisfaction and Use as outcomes of interest, the nature of training aspect is excluded from the present study.

It is expected that more effective training will lead to greater familiarity with the IT, thus enhancing the software developer's ability to predict his performance on software development tasks when using the IT. Stated in hypothesis form:

H11: As Software Developers' perceptions of Training effectiveness on a new IT increase, their perceived predictability with the new IT will increase.

2.6.6 Voluntariness of Use

Voluntariness of Use is defined as "the *degree* to which use of the innovation is perceived as being voluntary, or of free will [Moore 91, page 195]." In this study, we are consistent with IS research in treating Voluntariness of Use as a construct distinct from Champion Support. Although in subsequent data analysis we find a strong correlation between these two variables, we make no up-front assumptions about the relationship between them.

Moore and Benbasat [Moore 91] showed, through the testing and validation of their perceived voluntariness of use scale, that Voluntariness of Use is not a binary variable, as is often assumed in the literature; rather, their results show that there are varying degrees of voluntariness, as stated in their definition of the construct. Moore and Benbasat emphasize the importance of this variable in the study of diffusion of innovations since use of innovations within organizations may be either mandated or discouraged by organizational policy [Moore 91]. The latter policy takes the "freedom of choice" away from individual adopters.

Since mandated use of an IT takes away the freedom of choice from IT users, the following is hypothesized:

H9: As Software Developers' perceptions of Voluntariness of Use for a new IT increase, their perceived choice on the new IT will increase.

Further, it is expected that higher levels of perceptions of Voluntariness will lead to higher perceptions of software developer control over how the new IT is used in software development tasks. Thus:

H10: As Software Developers' perceptions of Voluntariness of Use for a new IT increase, their perceived process control on the new IT will increase.

2.7 Perceived IT Impacts

The research framework has identified two mediating variables to explain the impacts of Developer Involvement and Diffusion Environment on IT Diffusion Success: Perceived Control and Perceived IT Characteristics. In practice, however, software developers may also be motivated to use an IT innovation because the IT can measurably improve their effectiveness. Often, their effectiveness is measured in terms of software quality (e.g., the number of system defects) and/or productivity (e.g., lines of code delivered per person-month). IS and software engineering research has demonstrated benefits from the use of new software development techniques, including improved quality and productivity [Vessey 86, Walston 77, Baker 75, Linger 94]. Further, prior research has shown that proper training, and other environmental variables can improve developer quality and productivity [Ferguson 97]. Thus:

Proposition 5c: A significant, positive relationship exists between characteristics of the IT Diffusion Environment and software developers' Perceived IT Impacts.

Iivari studied software developer perceptions of quality and productivity; however, he examined these factors from the standpoint of how they were impacted from CASE tool use [Iivari 96]. The rationale was that CASE tools cannot be effective unless they are used. Thus, he examined the impacts of IT Use on Perceived IT Impacts.

This research takes an interactionist approach, positing that impacts from use of an IT impacts software developer's attitude towards subsequent use of the IT. Organizational behavior literature is the basis for interactionist theory, which states that there are interactions between people, their environment, and their behaviors [Schneider 83, Bandura 83]. Environment includes the work that must be performed (task) as well as the IT used to support the work [Green 95]. Part of interactionist theory states that there is feedback from the interaction between one's environment and one's behavior that impacts one's attitude. In the context of software development and the diffusion of software development techniques, this means that a software developer's attitude toward *future* IT Use is impacted by how he perceives his task performance when using the IT. Thus, the proposed research framework depicts a relationship between Perceived IT Impacts and Diffusion Success:

Proposition 6: A significant, positive relationship exists between software developers' Perceived IT Impacts and the Diffusion Success of the IT.

2.8 Summary of Current Literature

This section reviewed prior theory and research that guided the formation of the research framework, propositions, and hypotheses developed in this report. This research fits into the

overall research stream of diffusion of innovations and examines the factors that impact successful diffusion of software development techniques, including what *causes* these factors to impact diffusion. The following lines of reasoning are used to develop the relationships depicted in the research framework:

1. IS and DOI research establishes that IT Diffusion Success can be indicated by User Satisfaction with the IT and Use of the IT.
2. Diffusion of Innovations theory establishes that Perceived IT Characteristics impact IT Diffusion Success (Proposition 3).
3. Organizational Behavior Interactionist theory establishes Perceived Impacts as impacting IT Diffusion Success (Proposition 6).
4. Social Psychology theory establishes Perceived Control as a psychological factor that influences IT Diffusion Success (Proposition 1).
5. IS research establishes User Involvement as an external variable impacting Perceived IT Characteristics and Perceived Control (Propositions 2 and 4).
6. IS and Diffusion of Innovations research establishes Individual, Organizational, and Environmental Factors as impacting Perceptions (Propositions 5a, 5b, 5c).

Despite the contributions of the literature referenced above in forming the research framework, there are some shortcomings of the current literature that this research addresses. One shortcoming is that previous DOI research has focused on the diffusion of innovative software or hardware tools [Alexander 89, Iivari 96, Orlikowski 93, Chau 96]. However, recent innovations in software engineering have included improvements in software development processes and techniques. This research addresses this ever-growing suite of innovations by extending the concept of IT to include innovations in software development techniques.

Another shortcoming of current research has to do with the tasks and populations of interest in most existing IS studies on User Involvement and Satisfaction. Olson and Ives, Baroudi et al., and Doll and Torkzadeh all developed measures of the degree of end-user's participation in various stages of the software development cycle (e.g., system definition, design, code, test, installation, documentation, etc.) [Olson 81, Baroudi 86, Doll 94]. The approach of measuring participation in software development stages, however, is insufficient for this study as the specific software development activity of interest is product implementation, a subset of software development that has its own sub-stages (initiation, adoption, adaptation, acceptance, use, incorporation). Therefore, this research adapts items from existing User Involvement scales to develop a measure for user involvement suitable for involvement of software developers in the IT implementation process.

Similarly, there is a great deal of IS research with measures of end-user satisfaction as an indicator of IT success; however, none of these measures focus on software developers as the end-user of IT. Increased feelings of satisfaction have been linked with increased IT usage [DeLone 92, Baroudi 86]. Thus, in order to improve the use of effective software development techniques and, therefore, the effectiveness of software development, it is important to assess the satisfaction of developers. This research draws from existing, validated measures

of end-user satisfaction, and modifies them to address software developers as the population of interest.

A third shortcoming of current research is that there has been little work in an IS context that examines the impacts of perceived control on satisfaction and use. There has been no work in an IS context using the "broader" definition of perceived control, which includes behavioral control, choice, and predictability control. This broader definition is important when studying innovations used by software developers. Software developers are being asked to use software development innovations that take more creative control away from them, and simultaneously make their work more predictable. This research examines the broader construct of perceived control for its impacts on both satisfaction and use, which makes the findings more meaningful to the population of software developers. This research develops a measure of perceived control more comprehensive than existing measures found both in social psychology, management, and IS research.

A fourth shortcoming is that little research has examined the full linkages between user involvement, psychological factors, and IT success, as called for by Ives and Olson [Ives 84]. Doll and Torkzadeh present a model that includes psychological variables as mediators between user involvement and satisfaction, but do not test this mediating effect [Doll 89]. Hartwick and Barki present a model that includes the psychological variable user involvement as a mediator of the effect of user participation on use [Hartwick 94]. They subsequently found support for user involvement as a psychological variable. Baronas and Louis test the relationship between user involvement, perceived control, and satisfaction, but acknowledge that they were unable to verify that perceived control had actually been manipulated in their experiment [Baronas 88]. This research seeks to better understand the nature of the impact of involvement on IT success by examining the full linkages between user involvement, the psychological factor of perceived control, and IT success.

A final shortcoming of current research is that while there has been DOI research suggesting relationships between IT Diffusion Environment variables and perceptions [Orlikowski 93, Alexander 89, Moore 91], these linkages have not been tested empirically. This research explicitly tests these suggestions, using perceived control as the psychological perception of interest.

3 Research Methodology

This chapter describes the research study of the diffusion of software development techniques in software development organizations. In particular, the variables Developer Involvement in the Diffusion Process and IT Diffusion Environment Characteristics are examined for their impacts on Successful Diffusion of Software Development Techniques. This chapter discusses the research model that will guide the dissertation study, as well as the IT innovation that is used to test the research model. Next, hypotheses derived from the research model are presented. The chapter concludes with a discussion of the research design. This discussion includes a description of the survey instrument, procedures for administering the instrument, a description of the study sample, and a psychometric evaluation of the instrument.

3.1 Research Model

The research framework developed in Chapters 1 and 2 describes several key constructs and relationships as influencing the successful diffusion of software development techniques. Rather than study the entire framework in a single study, we have chosen to identify a subset of the research framework to study in this research. The chosen research model is shown in Figure 6.

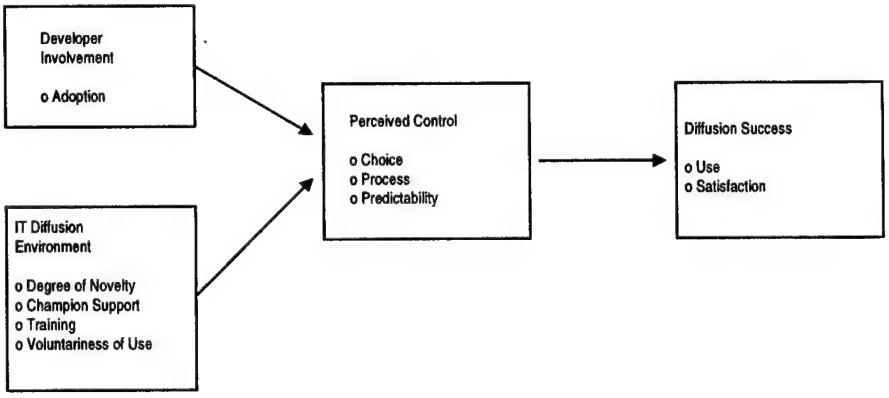


Figure 6: Research Model

Focusing on a subset of the framework enables a thorough study of a few of the research variables in a reasonable timeframe.

There are several explanations for the selected portion of the framework. First, the study of the constructs and relationships in the research model are deemed to be of immediate benefit to IS practice according to IS academics and practitioners (represented by the Software Engineering Institute). Second, the nature of the innovation chosen for this research (see next section for discussion of the innovation) is such that it may require the completion of a full development project in order to adequately assess Perceived Impacts. As such, it may be difficult to obtain much variation in data on Perceived Impacts (quality and productivity). Indeed, pilot results show very little variance in data on perceived quality impacts. Future research can focus on an innovation that is a more specific tool and/or technique, for which reliable data can be gathered over a shorter period of use. Finally, Perceived IT Characteristics have been widely studied in the IS Implementation and Diffusion of Innovations literatures. The Perceived Control construct has not been widely studied in the IS context; thus a significant theoretical contribution can be made to IS research.

Not all dimensions of constructs are chosen for inclusion in the research model. Specifically, the Application Domain and Tool Support variables are not included in the IT Diffusion Environment construct, and Developer Involvement in the *adaptation* of the IT is also excluded from the research model. The key rationale in these exclusions is again to allow the research to focus on those variables that are expected to contribute most to Perceived Control.

The software development technique that has been chosen to test the research model is the Personal Software Process[SM] (PSP[SM]) approach. The PSP approach is discussed next.

3.2 Personal Software Process (PSP)

The Personal Software Process approach is a "self-improvement process to control, manage, and improve the way [software engineers] work" [Humphrey 95]. PSP was developed by Humphrey in response to growing concern over how to move an organization beyond Level 2 of the Software Engineering Institute (SEI) Capability Maturity Model® (CMM®) framework [Humphrey 95]. Many of the practices described by the CMM framework address management practices [Ferguson 97]. However, Gibson notes that as organizations approach and move beyond Level 3 of the CMM framework, they find that further maturity is dependent on the improvement of individual software developer performance [Gibson 97]. Hence, the advent of the PSP approach.

PSP is not a software tool in the traditional sense of an IT. Rather, it is a disciplined approach to getting software developers to measure and analyze their work in order to improve their effectiveness as software engineers. Software developers who use the PSP approach apply quality principles such as defect elimination and incremental development in order to improve the quality of the software products they deliver. Individual programmers apply these

[SM] Personal Software Process and PSP are service marks of Carnegie Mellon University.
® Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark Office.

principles—thus, the degree to which organizations benefit from the PSP approach directly depends on the adoption and use of PSP by individual software developers.

Figure 7 depicts an overview of the sequence of steps to applying the PSP approach. The sequence is based on gradual introduction of disciplined methods in order to measure performance, analyze performance, and set goals to improve performance. The first phase, PSP0, focuses on developers learning to benchmark: i.e., understanding their current levels of productivity, quality, etc. PSP0.1 then introduces a Process Improvement Proposal (PIP) form to provide a structured way to record process problems, ideas for improvements, etc.

The second phase, PSP1, introduces the programmer to project planning as opposed to the project measuring emphasis in phase 1. It includes such principles as project sizing and test plan creation. PSP1.1 introduces developers to project management tasks such as basic task identification, task inputs and outputs, and task scheduling. Forms are included for recording all of these.

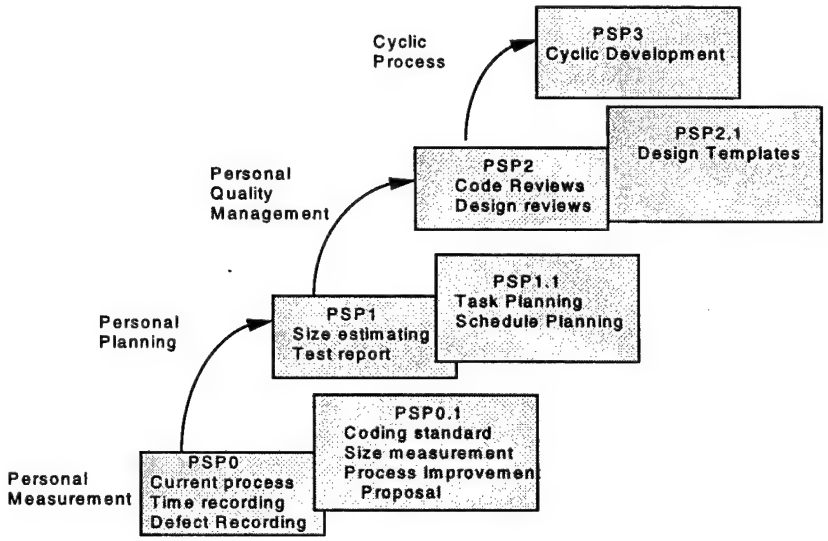


Figure 7: Summary of PSP Steps

The third phase, PSP2, focuses on defect management. Developers are introduced to the concept of design and code reviews. They are also given checklists to use in these reviews. PSP2.1 provides forms for developers to use in documenting internal and external program designs.

The fourth and final phase, PSP3, focuses on incremental development of systems. For each system increment, programmers are taught to design, code, compile, and test, then proceed to the next level in a top-down fashion. Each increment builds on a previous increment.

Case studies on the use of the PSP approach have reported several benefits to the individual software developer [Ferguson 97]. These benefits include improvements in planning and scheduling software development projects as a result of better estimation of work efforts; an improvement in software quality as a result of fewer program errors; and an overall reduction of software development time as a result of defect prevention and reduced rework required to fix errors. In a classroom environment, Humphrey has noted improvements in defect levels of individual engineers ranging from 58% to 72% [Humphrey 96]. He has also noted that with improvements in their ability to estimate their work efforts, software engineers can make more realistic commitments to management. Thus, the overall benefit to the individual software developer who adopts and uses the PSP approach is that he or she becomes a more productive and effective software engineer.

There are several advantages to using the PSP approach as the software development innovation in this study. One is the focus on individual software developers. PSP techniques are geared toward sustained use by individual developers in order to achieve increased effectiveness in software development. This enables us to maintain an individual level of analysis throughout the research data analysis. Another advantage of the PSP approach is that it is a fairly recent IT innovation, having been developed in the 1995 timeframe. As such, one can be fairly confident that PSP will be perceived as new by most software developers, therefore classifying PSP as an innovation.

A third advantage of using the PSP approach as the software development technique in this research is that PSP is a process, as opposed to a software product or tool. This allows us to research IT innovations using a broader definition of IT. Many non-software-based development techniques are being developed and implemented and can thus be studied using the research framework developed in this research [Linger 94]. Further, there is no need to be concerned about version changes or platform dependencies, as are common with software development tools.

A fourth advantage of the selection of the PSP approach is its fit with issues of Perceived Control. PSP is an innovation, which must be adopted on an *individual* level in order to be effective. PSP involves software developers taking significant, individual responsibility for improving the way they develop software; in other words, *the software developer is in control* of the software development improvement process. Thus, the selection of the PSP approach lends more support to the focus of the dissertation research on the Perceived Control construct, and use of PSP in particular can be expected to enhance feelings of personal control.

Finally, there is a tremendous amount of interest in the PSP approach. We have observed numerous books, papers, and personal conversations about PSP over the last year [Humphrey

95, Humphrey 96, Humphrey 97, Gibson 97, Ferguson 97]. Thus, the results uncovered in this research should be of interest to a wide audience of software development researchers and practitioners.

The next section of this chapter presents the research hypotheses that will be tested in this study. The hypotheses are derived from the research propositions identified in Chapter 2. The last section of this chapter presents the research methodology that will be used in the study.

3.3 Research Hypotheses

This section describes the research hypotheses about the impacts of Developer Involvement and IT Diffusion Environment on IT Diffusion Success. The hypotheses are described according to the outcomes that they relate to, either IT Diffusion Success, or Perceived Control.

3.3.1 IT Diffusion Success

The following proposition was developed in Chapter 2 to posit the relationship between Perceived Control and IT Diffusion Success:

Proposition 1: A positive, significant relationship exists between software developers' Perceived Control over their work when using an IT innovation, and the IT Diffusion Success.

Further discussed in Chapter 2 are the three dimensions of Perceived Control that are examined in this research: choice (ability to choose among alternative outcomes), process (ability to directly influence an event), and predictability (ability to foresee events). Baronas and Louis have suggested that enhanced perceptions of Choice are associated with increased levels of Satisfaction, although this link is not directly examined [Baronas 88]. This research examines this link with the following hypothesis:

H1: As Software Developers' perceptions of choice in using PSP increase, their level of Satisfaction with PSP will increase.

We have found no previous studies directly testing the relationship between Perceived Choice and Use. However, Baroudi et al. find that greater levels of Satisfaction with an IT lead to greater levels of IT Use [Baroudi 86]. Thus:

H2: As Software Developers' perceptions of choice in using PSP increase, their level of PSP Use will increase.

Tetrick and LaRocco studied the effects of predictability and process control on job satisfaction [Tetrick 87]. They found that when workers felt they had greater influence over the events in their work environment, they were more satisfied in their jobs. Thus, it is expected

that if software developers experience greater degrees of process control, then their Satisfaction with PSP will increase; that is:

H3: As Software Developers' perceptions of process control in using PSP increase, their level of Satisfaction with PSP will increase.

Taylor and Todd found a significant relationship between Perceived Behavioral Control (or process control) and IT Use [Taylor 95]. However, as noted in Chapter 2, their operationalization of Perceived Behavioral Control did not account for external constraints on behavior such as managerial behavioral control or constraints on behavior by the IT itself. This research operationalizes process control as freedom from constraints when using PSP to perform software development tasks. Consistent with Taylor and Todd and Mathieson, the following is hypothesized [Taylor 95, Mathieson 91]:

H4: As Software Developers' perceptions of process control in using PSP increase, their level of PSP Use will increase.

Regarding the relationship between Predictability and Use, Tetrick and LaRocco posited a direct association between predictability and job satisfaction, based on previous research in work stress health [Tetrick 87]. Although positive, moderate correlations are found between the two, confirmatory factor analysis suggests no direct relationship. However, somewhat low reliability of their Predictability scale (Cronbach's alpha = .66) suggests that subjects' true assessment of predictability may not have been adequately measured. Thus, this research continues to examine this direct association between perceived Predictability and Satisfaction in the following hypothesis:

H5: As Software Developers' perceptions of predictability using PSP increase, their level of Satisfaction with PSP will increase.

We have found no empirical studies directly examining the relationship between Predictability of events and Use. However, Fowler and Levine suggest this relationship in their case study of the diffusion of a new software development IT, Rate Monotonic Analysis scheduling (RMA) [Fowler 94]. Fowler and Levine present comments from developers of the adopting organization regarding their experiences with initial use of RMA. One reason given for continued use of the IT was their confidence in "always know[ing] how it will work." In other words, predictability in their work environment contributed to subsequent use. Therefore, it is hypothesized that greater perceptions of predictability in software development tasks when using PSP will be associated with greater levels of PSP Use:

H6: As Software Developers' perceptions of predictability using PSP increase, their level of PSP Use will increase.

Table 2 summarizes the hypothesized relationships between Perceived Control and IT Diffusion Success.

Propositions	Hypotheses	Independent Variables	IT Diffusion Success Outcomes	
			Satisfaction	Use
P1	H1, H2	Choice	+	+
	H3, H4	Process Control	+	+
	H5, H6	Predictability	+	+

Table 2: Hypothesized Relationships to IT Diffusion Success Variables

3.3.2 Perceived Control

The next set of hypotheses addresses factors that impact Perceptions of Control. The research model identifies five key variables as influencing Perceived Control. As discussed in Chapter 2, greater levels of User Involvement have been found in several studies to influence users' perceptions of control [Langer 83, Baronas 88]. Thus, the research framework includes the following relationship:

Proposition 2: A positive, significant relationship exists between Developer Involvement in the initiation, adoption, and adaptation of an IT innovation, and the software developers' Perceived Control over their work when using an IT innovation.

The initiation and adoption phases of the implementation process result in a decision to adopt or not adopt an IT innovation. The more involved a software developer is in this decision process, the more likely he is to perceive greater levels of choice in whether or not to adopt the IT. Thus, we can expect greater levels of involvement in the adoption process to result in greater levels of perceived choice in adoption. The hypothesis to be tested is as follows:

H7: As Software Developers are more involved in the PSP adoption process, their perceived choice in using PSP will increase.

The remaining four variables relate to aspects of the IT Diffusion Environment. The following proposition was offered in Chapter 2:

Proposition 5a: A significant relationship exists between characteristics of the IT Diffusion Environment and software developers' Perceived Control over their work when using the IT.

The first of the IT Diffusion Environment variables is Champion Support. Alexander defines champions as "individuals who advocate the adoption of an innovation," and have an ability to influence decisions [Alexander 89]. The degree to which a champion is able to influence decisions can be expected to influence the degree to which a developer feels free to make his own decision (choice) on adoption. Thus:

H8: As Software Developers' perceptions of degree of Champion Support for PSP increase, their perceived choice to use PSP will decrease.

Support for the above hypothesis is not an argument against Champion Support. To the contrary, champion and management support has been deemed beneficial in increasing IT use [Iivari 96, Chau, 96]. However, it may suggest the importance of balancing the potentially negative affect on Perceived Control with emphases on enhancing Perceived IT Characteristics and Impacts (other constructs depicted in the research framework), as well as incentives and other motivators (not included in the research framework) in order to ensure developer Satisfaction with and sustained use of PSP.

Another IT Diffusion Environment variable expected to influence Perceived Control is the degree to which the use of PSP is voluntary. Mandated use would be expected to result in low levels of perceived Voluntariness; voluntary use would be expected to result in higher levels of perceived Voluntariness. As indicated by Moore and Benbasat, mandated use of an IT takes away the freedom of choice from IT users. Thus, the following is hypothesized [Moore 91]:

H9: As Software Developers' perceptions of Voluntariness of PSP Use increase, their perceived choice to use PSP will increase.

Further, it is expected that higher levels of perceptions of Voluntariness will lead to higher perceptions of software developer control over how PSP is used in software development tasks. Thus:

H10: As Software Developers' perceptions of Voluntariness of PSP Use increase, their perceived process control over PSP will increase.

The next IT Diffusion Environment variable examined for its impact on Perceived Control is the adequacy of PSP Training. It is expected that more effective training will lead to greater familiarity with the IT, thus enhancing the software developer's ability to predict his performance on software development tasks when using the IT. Stated in hypothesis form:

H11: As Software Developers' perceptions of PSP Training increase, their perceived predictability using PSP will increase.

A final IT Diffusion Environment variable to be studied is the degree to which the use of PSP represents a new experience (i.e., degree of novelty). As discussed in Chapter 2, Orlikowski noted that radical change is often associated with ambiguity [Orlikowski 93]. Put in the context of the adoption of software development techniques, a technique (or suite of techniques) that represents a radical change to the software developer can lead to a work environment that is uncertain, unclear, and thus less predictable than more familiar techniques. This reasoning leads to the following hypothesis:

H12: As Software Developers' Degree of Novelty with PSP increases, their perceived predictability of using PSP will decrease.

The following table summarizes the hypothesized relationships to Perceived Control.

Propositions	Hypotheses	Independent Variables	Perceived Control Outcomes		
			Choice	Predictability	Process Control
P2	H7	Involvement	+		
P5a	H8	Champion Support	-		
	H9, H10	Voluntariness	+		+
	H11	Training		+	
	H12	Novelty		-	

Table 3: Hypothesized Relationships to Perceived Control

3.4 Research Design

This research is conducted as a field survey. The use of a field survey is appropriate for two reasons. First, the use of data from practicing software developers in multiple organizations increases the external validity of the results of the study. The results of the study can be more confidently applied across a wider population of software developers and software development organizations, an important aspect of applied research. Second, it would be more difficult to effectively manipulate the variables identified in the research model using an experimental design. One reason is that a relatively large number of independent variables (five) would have to be manipulated. Another reason is that manipulation of some of the independent variables would require significant time and/or monetary resources to accomplish. For example, training of subjects on PSP techniques would take several months to accomplish, along with the expense associated with securing an instructor, and time required of subjects to attend classes and complete assignments. For these reasons, a field survey is chosen as the design for this research.

The field survey is conducted in two stages: a pilot study and the full study. The purposes of the pilot study are to test the data collection procedure, validate the survey instrument, and provide data for initial tests of hypotheses. A convenience sample for the pilot study is drawn from the same population as that of the full study. Sampling procedures are described in more detail later in this chapter. Following the analysis of data from the pilot study as well as making required minimal changes to the survey instrument, the full study is conducted on a larger sample of PSP adopters.

3.4.1 Survey Instrument

Data to test the research hypotheses were collected via a questionnaire developed for this study. The questionnaire was pre-tested with IS professors, graduate students, and practitioners in order to ensure content validity. This pre-test preceded the study pilot test. The results of pre-testing were:

- shortening of the questionnaire to reduce the time needed to complete it to approximately 30 minutes; this "shortening" was accomplished via item deletion and reformatting of rating scales
- deletion of items that do not appear to adequately tap the variable being measured
- reformatting of rating scales such that a similar 7-point Likert scale was used for most questions
- modifications in item wording for clarity

The questionnaire contains eight sections: (1) General Information, (2) Involvement, (3) Organizational Environment, (4) Training, (5) PSP Characteristics, (6) Impacts, (7) Individual Use of PSP, and (8) Comments. With the exception of some items in section 7, the majority of items contained in the questionnaire use 7-point Likert scales ranging from either:

- "to no extent" to "to a great extent," or
- "strongly disagree" to "strongly agree"

depending on the wording of the items. The following sections describe each section of the questionnaire, including the items developed to measure each study variable, and the source of the item. A copy of the full questionnaire is included in Appendix A.

3.4.1.1 General Information

This section collects data on background variables. These variables are not included in the research model; however, they are included to ensure the respondent meets the criteria for inclusion in the study (e.g., use of PSP minimum of three months, maximum of two years), to check for potential covariation with the dependent variables, and to provide additional data for results interpretation. A summary of background variables is included in Table 4.

Item Description	Values
How long have you worked in the IS field?	Integer
Position title	Programmer Analyst Designer Project Manager Manager
Highest degree completed	High School Diploma Bachelors Masters Doctorate
Age	Integer
Gender	Male Female
Number of projects where PSP used	Integer
Length of time PSP used	Integer
Other techniques, tools, or methods used with PSP	Free-form text

Table 4: Summary of General Information Items

3.4.1.2 Involvement

The Involvement section of the survey instrument collects data on the degree to which the software developer respondent was involved in the initiation, adoption, and adaptation of PSP. Hartwick and Barki would refer to this variable as user participation [Hartwick 94]. Previous studies on user involvement/participation have been inconsistent in their operationalizations of the construct [Ives 84, Baroudi 86, Cavaye 95]. Olson, Baroudi, and Doll use different measures of the degree of end-user's participation in various stages of the software development cycle (e.g., system definition, design, code, test, installation, documentation, etc.) [Olson 81, Baroudi 86, Doll, 94]. The approach of measuring participation in software development stages is insufficient for this study as the specific software development activity of interest is product implementation, a subset of software development, which has its own sub-stages (initiation, adoption, adaptation, acceptance, use, incorporation).

Hartwick and Barki's three scales for User Participation provide the most extensive measure of the user involvement construct [Hartwick 94]. One of these scales looks at overall level of responsibility for the effort to implement the product, which is a broader view than looking at degree of participation in software development stages, and therefore more applicable to this research. Items from the Hartwick and Barki scales are adapted for the current study [Hartwick 94]. In addition, various items from the Doll and Torkzadeh scale that relate to system definition/feasibility are adapted since this stage of software development is somewhat similar to the initiation stage of implementation [Doll 94]. Finally, we develop items based on review of diffusion of innovations literature. The developer involvement items are averaged to produce an overall involvement score. A summary of the Developer Involvement measure is shown in Table 5.

Stage	Item	Source
	To what extent did you...	
Initiation	Have responsibility for estimating costs or benefits of implementing PSP?	[Hartwick 94]
	Participate in determining objectives of PSP use?	[Doll 94]
	Participate in assessing alternate ways to meet your development needs?	[Doll 94]
	Participate in identifying sources of information on PSP?	[Doll 94]
	Participate in initiating the effort to adopt PSP?	[Doll 94
	Participate in determining which development tools and techniques can address your development needs?	Researcher
Adoption	Lead the effort to adopt PSP?	[Hartwick 94]
	Have responsibility for requesting resources to support use of PSP?	[Hartwick 94]
Adaptation	How much did you participate in determining which parts of PSP to use in your organization?	Researcher

Table 5: Summary of Developer Involvement Measure

3.4.1.3 Organizational Environment

In the review of literature in Chapter 2, the IT Diffusion Environment construct of the research model was broken down into four aspects: Degree of Novelty, Champion Support, Training, and Voluntariness of Use. To include all of the items related to these four variables in one section of the questionnaire would have resulted in a very lengthy section; therefore, it was decided to place the Training-related items in a separate Training section of the questionnaire.

The Degree of Novelty variable discussed in Chapter 2 is said to refer to the respondent's assessment of the gap in skills required versus skills possessed to use PSP; that is, the respondent's assessment of his or her *individual ability to use PSP*. Because a subsequent section of the questionnaire contains items related to Individual Use of PSP variables, we felt that the Degree of Novelty items would fit more logically in that section, rather than the Organizational Environment section.

The placement of Training and Degree of Novelty variables in other sections of the survey results in the inclusion of two variables in the Organizational Environment section: Champion Support and Voluntariness of Use. Items for the Voluntariness of Use scale are drawn primarily from the Moore and Benbasat scale [Moore 91]. Moore and Benbasat report a Cronbach's alpha reliability coefficient of approximately .82 in two separate field tests of the Voluntariness scale [Moore 91]. Items for the Voluntariness of Use scale are averaged to produce an overall Voluntariness of Use score; similarly, items from the Champion Support scale are averaged to produce an overall Champion Support score. Most of the Champion Support items are taken from previously developed scales [Iivari 96, Chau 96]. A summary of the Voluntariness of Use and Champion Support scales is included in Table 6.

Variable	Item	Source
Champion Support	An IT champion can be defined as one who actively and vigorously promotes their personal vision for using IT, pushing a project over or around approval and implementation hurdles. Keeping this definition in mind, to what extent was there a champion for PSP in your organization?	Researcher
	Identify title and department of champion.	Researcher
	To what extent was the champion effective in promoting use of PSP?	Researcher
	How supportive was top non-IS management in the selection and implementation of PSP?	[Iivari 96]
	How supportive was IS management in the selection and implementation of PSP?	[Iivari 96]
	Management provided me with the time for training that I needed in order to use PSP effectively.	[Chau 96]
	Management provided me with the funding for training that I needed in order to use PSP effectively.	[Chau 96]
	I had easy access to people with the necessary expertise to help me make the transition to PSP.	[Chau 96]
	For making the transition to PSP, I felt I had a solid "network of support" in the form of knowledgeable colleagues, internal support personnel, and/or outside consultants.	[Chau 96]
Voluntariness of Use	My superiors expect me to use PSP.	[Moore 91]
	My use of PSP is voluntary (as opposed to required by my superior or job description).	[Moore 91]
	My supervisor does not require me to use PSP.	[Moore 91]
	Although it might be helpful, using PSP is certainly not compulsory in my job.	[Moore 91]
	Use of PSP is part of my performance plan and/or job description.	Researcher

Table 6: Summary of Organizational Environment Section

3.4.1.4 Training

There are six items on the questionnaire related to the Training aspect of the IT Diffusion Environment. Respondents are instructed to skip this section if they did not receive any training on PSP. Items are averaged to produce an overall Training score. Training items are summarized in Table 7.

Variable	Item	Source
Training	The training I received on PSP was adequate.	[Iivari 96]
	The quality of training I received on PSP was high.	[Iivari 96]
	There was not enough training for me on how to understand or use PSP.	[Goodhue 95]
	I have received the training I need to be able to use PSP effectively.	[Goodhue 95]
	Training on PSP was received prior to actual use of the techniques in a project.	Researcher
	Training on PSP was received at a time when I was simultaneously working on other assignments.	Researcher

Table 7: Summary of Training Measure

3.4.1.5 PSP Characteristics

The items in this section ask the respondent to assess the PSP approach based on their use of the PSP techniques. The primary variable being measured in this section is the respondents' overall satisfaction with PSP. DeLone and McLean (1992) observe that user satisfaction measures may be biased by user computer attitudes, and suggest that user satisfaction measures should also include measures of user attitudes in order to control for this potentially biasing effect [DeLone 92]. However, as Chau noted, since the population of interest in the study of diffusion of software development techniques is software developers, it can be assumed that this user group, on average, has a relatively positive attitude toward technology and computers
[Chau 96, Iivari 93]. Thus, a separate measure of developers' attitudes toward computers is not included in the survey. Items measuring developers' satisfaction with PSP are averaged to produce an overall satisfaction score. Table 8 summarizes the Perceived Characteristics items.

The *PSP Characteristics* section of the questionnaire also contains items measuring two Perceived IT Characteristics variables: Perceived Ease of Use and Perceived Usefulness. The Perceived IT Characteristics construct is included in the research framework described in Chapters 1 and 2, but is not included in the current research model. These data are collected with this study, but will be analyzed in subsequent stages of this research stream.

Variable	Item	Source
Satisfaction	The information provided by PSP provides sufficient information for improving software development effectiveness.	Doll and Torkzadeh [Doll 89]
	PSP provides information that is exactly what I need.	Doll and Torkzadeh [Doll 89]
	PSP meets my software development needs.	Doll and Torkzadeh [Doll 89]
Perceived Ease of Use	I found it easy to use PSP.	Davis [Davis 89a]
	The tools and techniques of PSP were clear and understandable.	Davis [Davis 89a]
	I found PSP to be flexible to implement.	Davis [Davis 89a]
	It was easy for me to become skillful at using PSP.	Davis [Davis 89a]
Perceived Usefulness	Using PSP improved my job performance.	Davis [Davis 89a]
	I feel that use of PSP was successful in transforming me into a more disciplined software engineer.	Researcher
	Using PSP made it easier to do my job.	Davis [Davis 89a]
	I found PSP useful in my job.	Davis [Davis 89a]

Table 8: Summary of PSP Characteristics Measure

3.4.1.6 Impacts of PSP

This section measures variables related to the software developers' perception of control and perception of impacts on their work using the PSP approach. These variables represent mediating variables in the research framework. Perceived Control is included in the research

model, and is operationalized as perceived choice, perceived process control, and perceived predictability. Items measuring these three variables are averaged to produce overall scores on the variables. As noted earlier in this chapter, Perceived Impacts is not included in the present research model; however, data are collected in this study and will be analyzed in future stages of this research. Table 9 summarizes the items for the PSP Impacts variables.

Variable	Item	Source
Perceived Choice	To what extent can you decide what parts of PSP you will use?	Tetrick and LaRocco [Tetrick 87]
	To what extent can you decide when you will use PSP techniques?	Tetrick and LaRocco [Tetrick 87]
	To what extent does the use of PSP allow you the opportunity for creativity in systems development?	Researcher
Perceived Process Control	There was a pre-specified sequence of steps that I was required to follow in my use of PSP.	Kirsch [Kirsch 96]
	I was required to follow existing standards and procedures in my use of PSP.	Kirsch [Kirsch 96]
Perceived Predictability	Since use of PSP, to what extend do unexpected results occur during systems development?	Tetrick and LaRocco [Tetrick 87]
	To what extent does the use of PSP techniques allow you to better predict the effort required for software development?	Researcher
	To what extent does the use of PSP techniques allow you to better predict the quality of software that you develop?	Researcher
Perceived Quality	Use of PSP has enhanced the functionality of applications that I build.	Iivari [Iivari 96]
	Use of PSP has decreased the number of errors in software products I build.	Iivari [Iivari 96]
	Software developed with PSP requires less maintenance.	Iivari [Iivari 96]
	Use of PSP has significantly improved my documentation of software products.	Iivari [Iivari 96]
	Use of PSP has made me more conscious of software quality.	Researcher
	On average, what percentage reduction in software errors, if any, have you experienced since implementing PSP techniques?	Researcher
Perceived Productivity	Use of PSP has greatly speeded up my development of new applications.	Iivari [Iivari 96]
	Use of PSP has made me more productive.	Iivari [Iivari 96]
	Use of PSP has significantly reduced the time I spend in software development.	Iivari [Iivari 96]
	On average, what percentage improvement in productivity, if any, have you experienced since implementing PSP?	Researcher

Table 9: Summary of PSP Impacts Measure

3.4.1.7 Individual Use of PSP

This section asks the respondents to respond to questions dealing with their experience in learning to use PSP (i.e., Degree of Novelty) and their current usage of PSP (Use). The individual items measuring Degree of Novelty are averaged to produce overall summary measures of these variables. Individual items measuring PSP Use are summed to produce an over-

all summary measure of these variables. Table 10 summarizes the Individual Use of PSP section.

Variable	Item	Source
Degree of Novelty	Initially, I felt that there existed a large gap between my existing skills and knowledge and those required by PSP	Chau [Chau 96]
	Major modification in our software development policies and procedures was necessary for PSP to truly fit in	Chau [Chau 96]
	The "language" of the PSP approach was not familiar to me	Chau [Chau 96]
	The underlying methodology of PSP was different from what we had used before	Chau [Chau 96]
Use	The proportion of projects I use PSP with: none 1-25% 26-50% 51-75% >75%	Iivari [Iivari 96]
	My use of PSP was/is: in initial project only in mostly small projects in a mixture of small and large projects in mostly large projects completely routine (in all my projects)	Iivari [Iivari 96]
	The phases that I typically use PSP principles in are: requirements specification software analysis software design implementation testing maintenance	Iivari [Iivari 96]
	How often do you use PSP?	Hartwick and Barki [Hartwick 94]

Table 10: Summary of Individual Use of PSP Measure

3.4.1.8 Comments

This final section of the survey provides respondents with a place for free-form comments. Respondents are asked to provide any additional information regarding their use of PSP that may help the researcher better understand their responses, or information that they feel is important to help the researcher understand their use of and reactions to PSP. This information may be of use in interpretation of results, and/or in identifying potential enhancements to the research framework.

3.4.2 Procedures for Survey Administration

The following procedure is used in the full study:

1. IS professionals who have adopted PSP were identified by the SEI and were contacted by the SEI to secure their participation in the study. The SEI licenses all PSP trainers; thus through these trainers, SEI has access to a majority of IS professionals who have been trained on PSP concepts. Those professionals who are IS managers or trainers were asked to distribute surveys to those software developers in their organization (or course)

who have been trained in and/or have used PSP. Those IS professionals who were not managers or trainers were contacted by the SEI (usually via a list-serv) and asked to participate in the study.

2. Surveys were given to the IS professionals who agreed to participate in the study. The number of surveys given to IS professionals was equal to the number of software developers to whom the IS professional agreed to distribute the surveys. The surveys were distributed via postal mail or via email attachments according to the preference of the recipient.
3. On the instructions to the survey, software developers were asked to complete the survey by a specific date (approximately three weeks from mailing) and return it in the stamped, self-addressed envelope accompanying the survey, or via email.
4. If questionnaires were not returned within one week after the date specified on the surveys, the appropriate contact person (either an IS manager or an individual) was contacted and asked to complete the survey(s) and return to us as quickly as possible.

For the pilot study, the SEI provided a list of IS professionals who are contacts in their organizations for PSP. A candidate pilot site was chosen from this list based primarily on convenience to the research team. After contacting the IS manager at the candidate site, a date for an onsite visit was arranged. On that date in July 1997, a one-hour structured interview was conducted with the IS manager followed by a half-hour structured group interview with the pilot software developers. The purpose of these structured interviews was to gain qualitative support for the relationships depicted in the research model, and to provide qualitative data that may prove useful in explaining unanticipated results in quantitative analyses. Finally, the software developers completed the surveys in our presence.

All questionnaires for the pilot study and all postal-mailed questionnaires for the full study were coded in such a way as to identify the organization that the software developer belongs to, and the number of the questionnaire within that organization. All emailed questionnaires for the full study were coded to indicate that they were sent and/or received via email. These codes were then mapped to email IDs.

3.4.3 Sample

Leonard-Barton and Deschamps suggest that successful implementation of new technology depends on acceptance by individuals [Leonard-Barton 88]. They further suggest that even after an organizational-level adoption decision, successful diffusion still depends on individual adoption decisions by target users. Thus, the unit of analysis in this study is individual software developers; the level of analysis is also at the individual level.

The population of interest in this study is software developers who have adopted a new software development technique (i.e., PSP) within the timeframe of three months to two years. A three-month minimum timeframe is specified in order to ensure that subjects have adequate time to have worked with PSP and form a more stable judgment. The two-year maximum timeframe is to ensure that the software development technique qualifies as an innovation [Rogers 95]. The sampling frame is given by the SEI as indicated in the Procedures for Sur-

vey Administration section, and consists of U.S. and non-U.S. software development organizations who have had employees trained in and/or using PSP within the last two years. The study sample is drawn from this sampling frame.

Nunnally recommends a sample size equivalent to 10 observations per model variable. For the purposes of this study, the minimum required sample size is investigated using power analysis with the following parameters [Nunnally 78, Cohen 88]:

- alpha level (probability of rejecting a null hypothesis when it is true) of 0.05
- effect size (magnitude or strength of the findings) of 0.40, the minimum suggested by Pedhazur and Schmelkin [Pedhazur 91]
- desired power (probability of detecting phenomena when it exists) of 0.80, as suggested by Cohen [Cohen 88]

The resulting required sample size is 83 subjects for the full study.

A total of 98 surveys were distributed between 11/19/97 and 3/20/98 via postal mail, and an additional 33 surveys were distributed during this same time period via email. Of these 131 surveys, 56 completed surveys were returned, resulting in a response rate of 42%. Additionally, ten surveys were distributed and completed in July, 1997 during the on-site pilot study. Because the survey instrument did not significantly change between 7/97 and 11/97 (see Survey Instrument Validation section), we decided to combine the two groups for data analysis. However, ANOVA tests did reveal significant differences in responses between the two groups on the model variables Degree of Novelty (prob>t = .002), Degree of Voluntariness (prob>t = .017), and Satisfaction with PSP (prob>t = .1). As a result, pilot data is removed when testing hypotheses involving these three variables.

As stated earlier in this section, the criteria for participation in the survey was that the software developer must have been using PSP for a minimum of three months and a maximum of two years. Of the 66 completed surveys, seven respondents indicate that their use of PSP is less than three months; one indicates that his use of PSP is longer than two years. This results in the removal of these eight respondents from subsequent data analysis. The resulting usable sample size is therefore 58. At an alpha level of .05 and an effect size of .4, this sample size provides a power level of 0.80 for hypothesis testing, and a power level of approximately 0.70 for the testing of the mediating effects of perceived control.

3.4.4 Survey Instrument Validation

This section describes the steps used to evaluate the psychometric properties of the survey instrument used in this study. The statistical analyses are performed using SPSS for Windows, Release 7.5.2.

3.4.4.1 Reliability

Reliability for the individual scales is assessed using Cronbach's alpha reliability coefficient [Cronbach 51]. Nunnally's suggested minimum of .70 reliability is desired [Nunnally 78].

Data collected during the pilot phase of the study was used to give an initial assessment of the reliability and validity of the survey instrument. After eliminating items that significantly reduce scale reliability, resulting alpha levels ranged from 0.72 to 0.95. These reliability levels exceed Nunnally's recommended minimum, and provide confidence that the survey scales do not contain a significant amount of random error. Thus, it was decided to keep the survey instrument intact, with the exception of the reformatting of responses for four questions (explained below) in order to provide more consistency in data analysis.

The pilot version of the research instrument contained four questions that required binary (yes/no) responses. Three of these were in Section II of the survey regarding developer involvement; one was in Section III of the survey regarding champion support. The remaining items on the surveys that were used in data analysis were rated on 7-point Likert scales. In order to avoid using different statistical measures for these four items, we reformatted the questions so that they could be rated on 7-point Likert scales, consistent with the other items.

Tables 11 through 15 present the results of the reliability assessments of the scales from the full study. In the initial reliability assessment of scales, Cronbach alpha levels for two of the 10 scales fall below the 0.7 desired level of internal consistency, as indicated by the reliability numbers in bold type in Table 11. Further analysis is conducted on all of the scales to see if these numbers can be improved. Scale reliabilities are assessed when each individual scale item is removed. By the removal of one of the items in six of the scales, reliabilities are improved. Asterisks in Table 11 indicate these items. Table 12 shows the resulting Cronbach alpha levels for the survey scales with items removed as indicated in Table 11. Finally, an additional reliability analysis is conducted for the Training and Use scales in order to further improve their levels of reliability. Again, this analysis involves looking at scale reliabilities after the removal of individual items. The result of this analysis is that by eliminating an additional item from each of these scales, reliability shows some improvement. Asterisks in Table 12 indicate these items. Table 13 shows the Cronbach alpha levels for the final version of survey scales with items removed as indicated in Tables 11 and 12. Finally, Table 14 gives a summary of the starting and ending reliability values for the scales. Consistent with the results found in the pilot study, the results from the full study show that the scales measuring the model constructs are reliable measures.

Variable/Scale	**Cronbach's Alpha**
Developer Involvement ... lead effort to adopt PSP ... have responsibility for requesting resources ... have responsibility for estimating costs/benefits ... determine objectives of PSP use ... assess alt ways of meeting needs ... identifying sources of info on PSP ... initiating the effort to adopt PSP ... determine which tools/technologies can address needs	.95
Champion Support ... champion for PSP	.90

... effectiveness of champion in promoting use ... supportiveness of non-IS mgmt in selection and implementation ... supportiveness of IS mgmt in selection and implementation ... time for training ... funding for training ... easy access to people to help me ... solid network of support	
Voluntariness ... superiors expect me to use PSP ... use of PSP is voluntary... ... supervisor does not require me to use PSP ... using PSP is not compulsory *... use of PSP is part of my performance plan...	.90
Training ... training I received was adequate ... quality of training was high ... not enough training on how to use PSP ... received training to use PSP effectively ... training timed appropriate to when I began to use PSP *... training received when working on other assignments	**.40**
Novelty ... large gap between existing skills and those required by PSP ... major modification in policies and procedures necessary ... "language" of PSP not familiar to me ... underlying methodology different from what was used before	.73
Choice ... can you decide what parts of PSP you will use ... can you decide when you will use PSP techniques *... does PSP allow creativity in systems development	.86
Process Control ... pre-specified sequence of steps that I was required to follow... ... required to follow existing standards...	.75
Predictability *... do unexpected results occur during systems development ... does use of PSP allow you to better predict effort ... does use of PSP allow you to better predict quality of software	**.48**
Satisfaction *... PSP info provides sufficient info for improving effectiveness ... PSP is exactly what I need ... PSP meets my software development needs	.86
Use ... proportion of projects I use PSP with... ... use of PSP was/is ... *... phases I typically use PSP principles in.. ... how often do you use PSP	.77

Table 11: Reliability Coefficients for Original Scales

Variable/Scale	Cronbach's Alpha
Developer Involvement ... lead effort to adopt PSP ... have responsibility for requesting resources ... have responsibility for estimating costs/benefits ... determine objectives of PSP use ... assess alt ways of meeting needs ... identifying sources of info on PSP ... initiating the effort to adopt PSP ... determine which tools/technologies can address needs	.95
Champion Support ... champion for PSP ... effectiveness of champion in promoting use ... supportiveness of non-IS mgmt in selection and implementation ... supportiveness of IS mgmt in selection and implementation ... time for training ... funding for training ... easy access to people to help me ... solid network of support	.90
Voluntariness ... superiors expect me to use PSP ... use of PSP is voluntary... ... supervisor does not require me to use PSP ... using PSP is not compulsory	.91
Training ... training I received was adequate ... quality of training was high ... not enough training on how to use PSP ... received training to use PSP effectively *... training timed appropriate to when I began to use PSP	.74
Novelty ... large gap between existing skills and those required by PSP ... major modification in policies and procedures necessary ... "language" of PSP not familiar to me ... underlying methodology different from what was used before	.73
Choice ... can you decide what parts of PSP you will use ... can you decide when you will use PSP techniques	.93
Process Control ... pre-specified sequence of steps that I was required to follow... ... required to follow existing standards...	.75
Predictability ... does use of PSP allow you to better predict effort ... does use of PSP allow you to better predict quality of software	.84
Satisfaction ... PSP is exactly what I need ... PSP meets my software development needs	.91
Use ... proportion of projects I use PSP with... ... use of PSP was/is ... *... how often do you use PSP	.84

Table 12: Reliability Coefficients for Shortened Scales

Variable/Scale	Cronbach's Alpha
Developer Involvement ... lead effort to adopt PSP ... have responsibility for requesting resources ... have responsibility for estimating costs/benefits ... determine objectives of PSP use ... assess alt ways of meeting needs ... identifying sources of info on PSP ... initiating the effort to adopt PSP ... determine which tools/technologies can address needs	.95
Champion Support ... champion for PSP ... effectiveness of champion in promoting use ... supportiveness of non-IS mgmt in selection and implementation ... supportiveness of IS mgmt in selection and implementation ... time for training ... funding for training ... easy access to people to help me ... solid network of support	.90
Voluntariness ... superiors expect me to use PSP ... use of PSP is voluntary... ... supervisor does not require me to use PSP ... using PSP is not compulsory	.91
Training ... training I received was adequate ... quality of training was high ... not enough training on how to use PSP ... received training to use PSP effectively	.77
Novelty ... large gap between existing skills and those required by PSP ... major modification in policies and procedures necessary ... "language" of PSP not familiar to me ... underlying methodology different from what was used before	.73
Choice ... can you decide what parts of PSP you will use ... can you decide when you will use PSP techniques	.93
Process Control ... pre-specified sequence of steps that I was required to follow... ... required to follow existing standards...	.75
Predictability ... does use of PSP allow you to better predict effort ... does use of PSP allow you to better predict quality of software	.84
Satisfaction ... PSP is exactly what I need ... PSP meets my software development needs	.91
Use ... proportion of projects I use PSP with... ... how often do you use PSP	.87

Table 13: Reliability Coefficients for Final Scales

Variable/Scale	No. of Items (original)	Cronbach's Alpha	No. of Items (reduced)	Cronbach's Alpha
Developer Involvement	8	.95	8	.95
Champion Support	8	.90	8	.90
Voluntariness	5	.90	4	.91
Training	6	.40	4	.77
Novelty	4	.73	4	.73
Choice	3	.86	2	.93
Process Control	2	.75	2	.75
Predictability	3	.48	2	.84
Satisfaction	3	.86	2	.91
Use	4	.77	2	.87

Table 14: Reliability Coefficients for Scales

Table 15 shows how the reliability measures for the scales used in this study compare to reliability levels of previously developed measures from which items in the present scales were drawn. These results show that with one exception (the Training scale), the scales in the current study have reliabilities comparable to or better than reliability levels reported for existing scales.

3.4.4.2 Validity

Construct (convergent) validity assessment of the scales assesses the degree to which the items are measuring the target constructs. A factor analysis using principal axis factoring extraction is used to assess convergent validity. All items in a scale would be expected to load highly on a single factor in order to demonstrate high construct validity. Ideally, the approach to demonstrating convergent validity would be to include all items from the survey instrument in a factor analysis and validate that all items for a scale load on a single factor, resulting in 10 factors extracted. Using the 9:1 rule of thumb ratio, this approach would require a sample size of approximately 342. Since the sample size in this study is only 58, the "all items" approach is not statistically viable. Thus, the approach chosen for this study is to perform factor analyses on each scale, extracting one factor, and ensuring all scale items load on a single factor. Minimum factor loadings of 0.50 are desired. Because only one factor was extracted in each factor analysis, no rotation was performed.

Variable/Scale	Existing Scale	Number of Items	Existing Scale Reliability	Current Study Scale Reliability
Developer Involvement	Hartwick and Barki [Hartwick 94]	6	.80	.95
	Doll & Torkzadeh [Doll 94]	8	.96	-
	Iivari [Iivari 96]	2	.76	-
Champion Support	Iivari [Iivari 96]	2	.66	.90
	Chau (1996)	4	.84	-
Voluntariness	Moore and Benbasat [Moore 91]	2	.82	.91
	Iivari [Iivari 96]	2	.88	-
Training	Iivari [Iivari 96]	2	.84	.77
	Goodhue & Thompson [Goodhue 95]	2	.74	-
Novelty	Chau [Chau 96]	5	.74	.73
Choice	n/a	-	-	.93
Process Control	n/a	-	-	.75
Predictability	Tetrick & LaRocco [Tetrick 87]	3	.66	.84
Satisfaction	Doll & Torkzadeh [Doll 94]	12	.92	.91
Use	Iivari [Iivari 96]	8	.88	.87

Table 15: Comparison of Previous and Current Scale Reliabilities

In each individual factor analysis, all items load on a single factor. There is one item in the Training scale that loads at 0.48; all other items load on their respective factors at levels between 0.53 and 0.93. Thus, we can conclude that the scales exhibit strong convergent validity.

An additional assessment of construct validity is done for the perceived control construct. Since this construct is being examined from three dimensions (perceived choice, perceived process control, and perceived degree of predictability), it is proper to perform a factor analysis to determine if, in fact, the perceived control items will load on three distinct dimensions. First, a correlation analysis is performed to determine if there is a high degree of correlation between the six perceived control items. The result of this analysis shows that an item from the perceived process control scale is moderately correlated with the two items from the perceived choice scale. Additionally, an item from the perceived choice scale is moderately correlated with the two items from the predictability scale. Both varimax and oblique rotations yield similar factor structures. However, the results of item correlations suggest that moderate factor correlations may be present. As such, an oblique rotation is used in the factor analysis. Subsequent results of the factor analysis using an oblique rotation did confirm moderate correlation between the three factors. Table 16 shows the factor loadings that result from the factor analysis, which confirms the existence of three dimensions of perceived control.

The general rule of thumb for confidence in factor analysis results is that there be a minimum of three indicators per factor. The present analysis uses only two indicators per factor. However, when there is correlation between at least two of the factors, then two indicators per factor provide sufficient confidence in results. The above results of the reliability and validity assessments demonstrate sufficient validity to proceed with the testing of the research hypotheses. The remaining data analysis activities are described in the next chapter.

Variable	Item	Factor 1	Factor 2	Factor 3
Choice	IMP4	**.99**	.01	-.03
	IMP5	**.85**	-.03	-.01
Process	ORG15	-.13	**.73**	**-.08**
	ORG16	.07	**.84**	.06
Predictability	IMP2	-.03	-.11	**-.87**
	IMP3	.07	.12	**-.83**

Table 16: Factor Analysis for Perceived Control Variables

4 Data Analysis

A research framework for the diffusion of software development techniques has been proposed. This framework is grounded in established theory in IS, Social Psychology, Diffusion of Innovations, and Software Engineering. The framework is a causal model, which shows relationships between theoretical constructs that affect the successful diffusion of software development techniques. Data analysis techniques cannot "prove" the theory depicted in the causal model; proof or support for the theory represented by the causal model has been argued in the preceding chapters on the basis of the theories upon which the model is based [Popper 59, Pedhazur 82]. However, the data analysis can shed light on whether or not the causal model is consistent with the data collected in the study. If the model is consistent with the data, then the theory has survived Popper's disconfirmation test [Popper 59].

One technique that is suitable for the analysis of a causal model is path analysis [Blalock 71, Pedhazur 82, Tait 88]. Path analysis allows the examination of direct effects of one variable on another, as well as indirect effects of one variable on another through one or more intervening (mediating) variables. Path analysis makes use of a series of multiple regressions where path coefficients are used to assess the strength of the relationships depicted in the causal model. Thus, the magnitude, direction, and significance of the path coefficients can be used to test the research hypotheses. Path analysis has been used to analyze the research model in this report and test the corresponding hypotheses. For convenience, the research model is replicated in Figure 8.

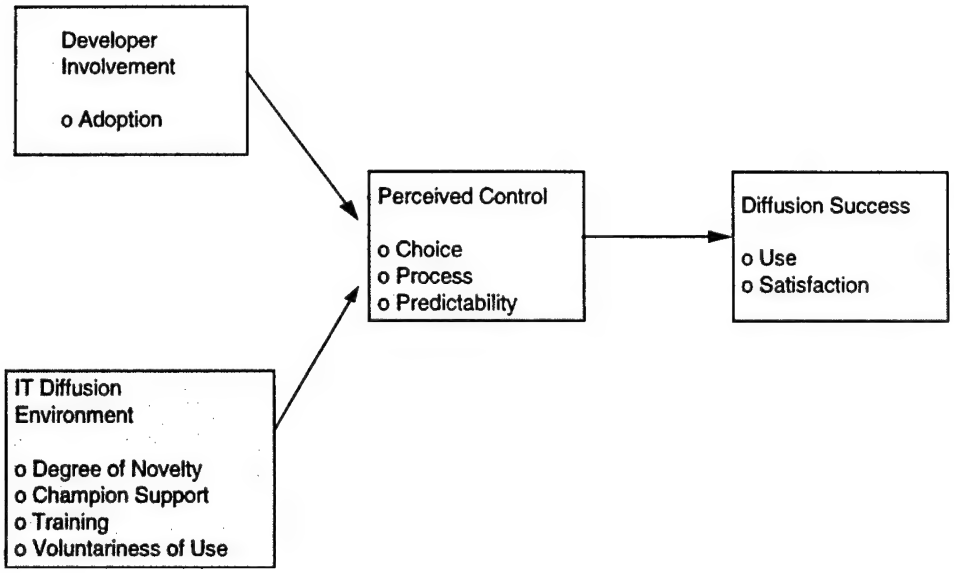


Figure 8: Research Model

The next section presents descriptive statistics on the study sample and on the study variables. The remaining sections of this chapter discuss the results of the multiple regression analyses, as well as the implications of these results.

4.1 Descriptive Statistics

Section 1 of the survey instrument collects demographic information on survey respondents, as well as general information on PSP use. In addition, we tracked the origin of the survey respondents as either U.S. or non-U.S.. Table 17 summarizes this descriptive data. The table shows that the majority of survey respondents are approximately 34-year-old male programmers from U.S. organizations, who hold either bachelors or masters degrees, and who have worked in the IS field for eight years. Further, descriptive data show that most of these software engineers have used PSP for approximately one year.

Variable	Value
Gender o male o female	 79 % 21 %
Age	Mean = 33.6 Range= 21 - 60
Position o analyst o designer o manager o programmer o project manager o other	 12 % 21 % 5 % 47 % 14 % 1 %
Highest Degree Held o high school o bachelors o masters o Ph.D.	 3 % 47 % 47 % 3 %
Years IS Experience	Mean = 8.4 Range= 1 - 22
Months of PSP Experience	Mean = 12.4 Range= 3 - 24
Origin of Respondent (organization) o US o non-US	 88 % 12 %

Table 17: Summary of Demographic Information

Descriptive data on the model variables are presented in Table 18. With the exception of the PSP Use items, which are summed, items for study variables are rated on a 7-point scale from low to high and then averaged to produce an overall variable score. PSP Use items are summed; the lowest possible score would be zero, indicating no use of PSP beyond PSP training. The highest possible PSP Use score would be 12, indicating that PSP is used on over 75% of the software developers' projects, and that the use of PSP is extremely frequent.

The data show that in general, involvement of software engineers in the adoption of PSP is low (mean = 3.0). On the contrary, quality and adequacy of PSP training is deemed to be fairly high (mean = 5.86). It appears that it is fairly common for PSP to receive strong champion support in organizations, as indicated by the mean score of 5.05. The mean score of 7.72 for PSP Use out of a possible 12.0 indicates that use of PSP is not yet completely routine, and that PSP techniques are not yet used throughout the software development life cycle.

Variable	Mean	Range	Standard Deviation
Developer Involvement	3.0	1.0 - 7.0	1.80
Champion Support	5.05	1.5 - 7.0	1.25
Voluntariness	4.12	1.0 - 7.0	1.93
Training	5.86	3.25 - 7.0	0.78
Novelty	4.0	1.0 - 7.0	1.48
Perceived Choice	5.03	1.0 - 7.0	1.60
Perceived Process Control	5.81	1.0 - 7.0	1.87
Perceived Predictability	4.16	1.0 - 6.5	1.40
Satisfaction with PSP	4.65	1.0 - 7.0	1.34
Use of PSP	7.72	1.0 - 12.0	3.40

Table 18: Descriptive Statistics of Model Variables

Exploring descriptive data on PSP Use further, Table 19 shows that PSP has been used by software developers on an average of two projects (mean = 2.4). Over 75% of these projects have either been mission-critical projects, or have been a mixture of critical and non-critical projects. Over one-quarter of the software developers surveyed currently use PSP techniques on all their software development projects.

Variable	Value
Number of PSP Projects	Mean = 2.4 Range= 0 - 18
How PSP Used: o critical projects only o non-critical projects only o mixture of critical and non-critical projects	 39 % 24 % 37 %
Percentage of Projects using PSP o none o 1 - 25 % o 26 - 50 % o 51 - 75 % o > 75 %	 12 % 28 % 9 % 18 % 33 %
PSP Project Types: o proof of concept only o small projects only o mixture of large and small projects o large projects only o all projects	 32 % 7 % 26 % 9 % 26 %
Project Phases where PSP Used o requirements specification o analysis o design o implementation o testing o maintenance	 32 % 39 % 73 % 83 % 62 % 20 %

Table 19: Descriptive Data on PSP Use

4.2 Model and Hypotheses Testing Strategy

The strategy for assessing the viability of the theoretical relationships presented in the research model and derived hypotheses is to perform a series of multiple regressions. The variables used in the multiple regression equations correspond to those depicted in the theory-based research model developed in Chapter 2 (see Figure 8). As noted in Chapter 3, values for the model independent variables are derived by averaging the scores for the survey items that measure them. This averaging approach has the disadvantage of ignoring potential measurement errors during data analysis, which means that potential model relationships may be hidden. This concern can be addressed in future studies by using a larger sample size and a more powerful statistical analysis technique such as structural equation modeling, which allows for more precise estimation of measurement errors, and thus a more complete analysis of model relationships considering these error terms.

Correlation between independent variables is checked via a correlation analysis, summarized for all model variables in Table 20. Correlations between most of the independent variables are low, suggesting that the variables are distinct and independent. However, the high degree of negative correlation between Champion Support and Voluntariness suggests that the results of the regression model, which includes these two independent variables, may produce spurious results. However, the researchers have retained these two variables as they are distinct variables grounded in previous theory. Statistical techniques are used and described later in this chapter that enable us to understand the effects of these two variables on dependent variables without the risk of spurious results.

	Involvement	**Champion**	**Voluntariness**	**Training**	**Novelty**
Involvement	1.000				
Champion	-0.110 (p=.410)	1.000			
Voluntariness	0.227 (p=.087)	**-0.596** (p=.000)	1.000		
Training	0.043 (p=.751)	0.005 (p=.972)	0.039 (p=.775)	1.000	
Novelty	0.019 (p=.888)	0.122 (p=.361)	-0.221 (p=.096)	0.182 (p=.180)	1.000

Table 20: Pearson Correlations Between Independent Variables

Figure 9 is a depiction of the research hypotheses derived from the research model and described in Chapter 3. Support for the study hypotheses is assessed by examining the direction and significance of the standardized beta coefficients for each path in Figure 9.

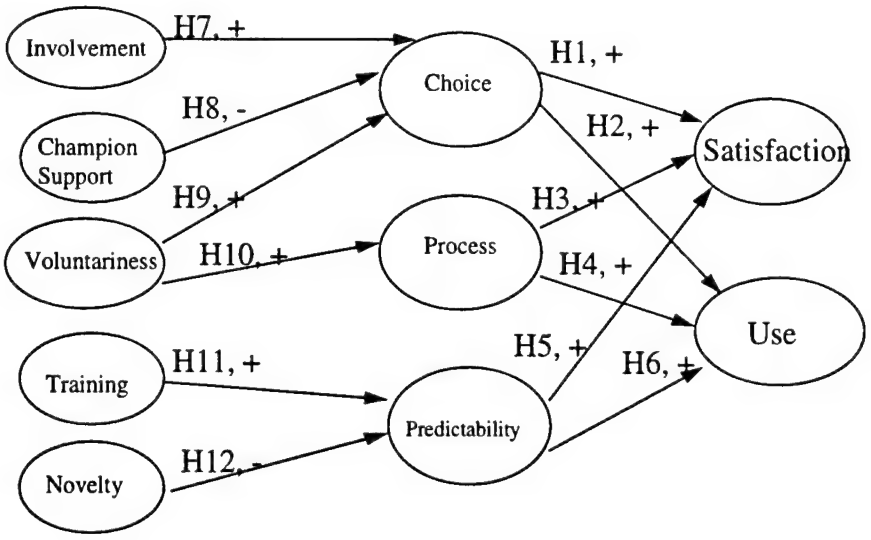


Figure 9: Summary of Research Hypotheses

The number of regression equations needed to test study hypotheses is equal to the number of dependent variables depicted in Figure 9, which totals five regressions. In addition, in order to test the implication of the model that the impacts of the independent variables on the IT Diffusion Success variables are only through the mediating effects of the perceived control variables, two additional regressions are run testing these direct paths. Table 21 shows the regression models used to test study hypotheses and mediating effects of Perceived Control. A mapping of regression runs to hypotheses tested is presented in Table 22.

Regression Number	Regression Equation
1	$SAT = B_0 + B_1PREDICT + B_2PROCESS + B_3CHOICE$
2	$USE = B_0 + B_1PREDICT + B_2PROCESS + B_3CHOICE$
3	$CHOICE = B_0 + B_1INVOLVE + B_2VOLUN + B_3CHAMPION$
4	$PROCESS = B_0 + B_1VOLUN$
5	$PREDICT = B_0 + B_1TRAINING + B_2NOVELTY$
6	$SAT = B_0 + B_1TRAINING + B_2NOVELTY + B_3INVOLVE + B_4CHAMPION + B_5VOLUN$
7	$USE = B_0 + B_1TRAINING + B_2NOVELTY + B_3INVOLVE + B_4CHAMPION + B_5VOLUN$

Table 21: List of Regression Equations

Regression Number	Dependent Variable	Hypotheses
1	Satisfaction	H1, H3, H5
2	Use	H2, H4, H6
3	Choice	H7, H8, H9
4	Process	H10
5	Predictability	H11, H12
6	Satisfaction	Mediating Effects
7	Use	Mediating Effects

Table 22: Mapping of Regression Models to Hypotheses Tested

The regression equations include error terms that represent the difference between observed and predicted values of the dependent variables. Regression models are based on the following assumptions about the error term:

- the average value of the error terms is zero
- the variance of the error terms is constant
- the distribution of error terms is normal
- error terms are independent of each other

The validity of inferences made from the results of these regression analyses will depend on whether the above assumptions are satisfied. To this end, the above four assumptions are checked for each of the regression models. The details of the analyses and the plots are found in [Green 98].

4.2.1 Mean of Errors Is Zero

To check this assumption, partial residual plots of each independent variable with the dependent variable are produced and analyzed for discernible trends or patterns. Because no patterns or trends are observed, we can conclude that this assumption is satisfied.

4.2.2 Variance of Errors Is Constant

Scatterplots of residuals against predicted values of the dependent variables are produced and analyzed for extreme variability or heteroscedascity. Because there is no apparent tendency for the variances to increase as the dependent variables increase, we can conclude that this assumption is satisfied.

4.2.3 Distribution of Errors Is Normal

Normal probability plots are generated to compare the distribution of dependent variable residuals to a normal distribution. In viewing these plots, it is apparent that the distributions of the error terms are approximately normal. Thus, we can conclude that this assumption is satisfied.

4.2.4 Error Terms Are Independent

Because we have not collected time-series data (data recorded over time), we can safely assume that the residual terms are independent.

4.2.5 Outliers and Influential Observations

In order to detect the existence of outliers and influential observations, a casewise analysis of residuals is generated in SPSS for each of the regression runs. This analysis shows, for each observation, the predicted and actual values of the dependent variable, as well as the difference between the two (i.e., the residual). Influential observations are deemed as those observations for which the residual was more than three standard deviations from the mean. The result of this analysis is that no observation is found to be influential. Thus, no observations are removed from the data set as a result of this analysis.

The next section presents the results of the multiple regression analyses.

4.3 Results and Discussion

Table 23 summarizes the results of running the five regression equations that test the study hypotheses. Note that the sample sizes for each equation differ from 58 (the study sample size) due to (1) the eliminating of pilot data (N=9) for regressions dealing with Satisfaction, Degree of Novelty, and Degree of Voluntariness; and (2) missing data for dependent variables. A significant amount of variance in Satisfaction with PSP, Perceived Choice, and Perceived Process Control is explained by model variables (adjusted R2 = .28, .41, and .37 respectively). Less variance is explained for the PSP Use and Perceived Predictability dependent variables (adjusted R2 = .09 and .05 respectively), which indicate that there may be other variables not included in the research model that may better explain IT Use and Perceived Predictability. The next two sections discuss the results of individual hypothesis testing.

Reg. #	Regression Equation	N	Sig. F	Adj. R^2
1	$SAT = B_0 + B_1PREDICT + B_2PROCESS + B_3CHOICE$	44	.001	.28
2	$USE = B_0 + B_1PREDICT + B_2PROCESS + B_3CHOICE$	53	.055	.09
3	$CHOICE = B_0 + B_1INVOLVE + B_2VOLUN + B_3CHAMPION$	46	.000	.41
4	$PROCESS = B_0 + B_1VOLUN$	48	.000	.37
5	$PREDICT = B_0 + B_1TRAINING + B_2NOVELTY$	42	.134	.05

Table 23: Summary of Regression Analyses

4.3.1 Relationship Between Perceived Control and IT Diffusion Success

Six hypotheses examine the relationship between Perceived Control and IT Diffusion Success. They are as follows:

H1: As Software Developers' perceptions of choice in using PSP increase, their level of Satisfaction with PSP will increase.

H2: As Software Developers' perceptions of choice in using PSP increase, their level of PSP Use will increase.

H3: As Software Developers' perceptions of process control in using PSP increase, their level of Satisfaction with PSP will increase.

H4: As Software Developers' perceptions of process control in using PSP increase, their level of PSP Use will increase.

H5: As Software Developers' perceptions of predictability when using PSP increase, their level of Satisfaction with PSP will increase.

H6: As Software Developers' perceptions of predictability when using PSP increase, their level of PSP Use will increase.

Table 24 shows the results of testing the hypotheses relating Perceived Control independent variables to IT Diffusion Success dependent variables. Two of the three hypotheses relating Perceived Control to Developer Satisfaction are supported. Specifically, an increase in a developer's sense of Choice in using PSP is positively and significantly associated with the developer's sense of Satisfaction with using PSP (Hypothesis H1). This result shows that the more that developers are given the freedom to choose when to apply PSP in their work and what parts of PSP to apply, the more satisfied they are in using PSP. Further, a developer's ability to better predict his work outcomes when using PSP is positively and significantly associated with the developer's sense of Satisfaction with using PSP (Hypothesis H5). Developers are more satisfied with using PSP when they are able to better estimate their work efforts and subsequent work quality by using PSP.

Dependent Variable	Independent Variable	Hypothesis	Hypothesized Direction	Beta	Sig.
Satisfaction	Choice	H1	+	**.316**	**.028**
	Process	H3	+	-.221	.103
	Predict	H5	+	**.349**	**.015**
Use	Choice	H2	+	.005	.974
	Process	H4	+	-.144	.300
	Predict	H6	+	**.339**	**.020**

Table 24: Results of Testing Perceived Control Hypotheses

The result of testing Hypothesis H3 is quite interesting: the data suggest that the more control one has over how to use PSP, the *less* Satisfied one is with using PSP. This result is seen by the negative and marginally significant Beta coefficient in the path relating Perceived Process Control to Satisfaction (Beta = -.221, p=.103). In other words, the more standards and structure that are emphasized in use of PSP, the more Satisfied the developer is in using PSP. This result contradicts the stated hypothesis as well as some previous research relating process control to satisfaction [Tetrick 87]. However, this result is consistent with prior research examining the effects of process control on IS development team performance (Henderson and Lee, 1992). Henderson and Lee found that decreases in one's personal process control resulted in higher levels of IS team performance [Henderson 92]. While the Henderson and Lee study and the current study examine different outcomes (performance and satisfaction, respectively), the studies have in common the fact that they examine process control from the standpoint of IS developers [Henderson 92]. The findings in the current study as well as Henderson and Lee suggest that there may be differences between IS personnel and/or tasks, and the personnel/tasks studied in previous research on process control, which may account for the differences in the effect that process control has on Satisfaction [Henderson 92]. For example, IS tasks have been noted in previous research to represent complex, unstructured tasks [Khalil 89, Vessey 98]. By enforcing standards and structure, a software development technique can bring structure to a previously unstructured task environment, thereby reducing overall task complexity. Thus, contrary to current social psychology literature on perceived process/behavioral control, the present research suggests that in a complex task environment such as software development, decreases in personal control as a result of the use of a software development technique result in greater Satisfaction with using that technique.

Alternatively, there may be other factors not accounted for in this study that may explain the negative relationship between Perceived Process Control and Satisfaction. For example, Greenberger and Strasser (1996) develop a model of Personal Control that accounts for differences in Perceived Control (i.e., behavioral control possessed) and Desired Control (i.e.,

behavioral control desired). These two variables may interact to affect a developer's Satisfaction with the software development technique. Indeed, the Theory of Planned Behavior (TPB), discussed in Chapter 2, includes a similar interaction in its definition of Perceived Behavioral Control [Ajzen 91]. Specifically, TPB defines Perceived Behavioral (process) control as beliefs regarding the extent to which internal and external factors impede performance of a behavior [Taylor 95]. Perceived Behavioral Control is represented as the product of the Behavioral Control possessed and the importance of Behavioral Control to the individual. The differences in IS vs. non-IS personnel, and potential interaction between perceived and desired process control should be examined in future research as possible explanations for IS developers' Satisfaction with increased controls over how they use software development innovations.

Results of the hypotheses relating Perceived Control to Use are mixed. Table 24 shows that perceived Predictability has a positive and significant relationship to PSP Use, as demonstrated by the .339 Beta coefficient of this variable (p=.02). This result suggests that software developers are more likely to use an innovative software development technique when that technique improves their ability to predict software development work outcomes. However, a developer's perceptions of Choice and Process Control do not appear to have significant relationships with the developer's Use of PSP. Insignificant Beta values (at an alpha level of .1) for both the process and choice variables do not provide support for hypotheses H2 and H4. Additional analyses to better understand this outcome will be undertaken and discussed in the next section of this chapter.

The foregoing results give strong support to the assertion that a developer's sense of Perceived Control over his work when using an IT innovation has a significant relationship to the developer's Satisfaction with the IT innovation. The above results also give some support to the assertion that a developer's sense of Perceived Control over his work when using an IT innovation has a significant relationship to the developer's subsequent Use of the IT innovation. Overall, there is moderate support for the study proposition that a developer's sense of Perceived Control has a significant relationship with IT Diffusion Success. Because these results suggest that enhancing Perceived Control is important to the successful diffusion of IT innovations, it becomes important to understand those factors that influence Perceived Control. The next set of hypotheses examines factors that are believed to influence software developers' perceptions of control.

4.3.2 Relationship Between Involvement and Environment Factors, and Perceived Control

Six hypotheses examine the relationship between Involvement and Environment Factors, and Perceived Control. They are as follows:

H7: As Software Developers are more involved in the PSP adoption process, their perceived choice in using PSP will increase.

H8: As Software Developers' perceptions of degree of Champion Support for PSP increase, their perceived choice to use PSP will decrease.

H9: As Software Developers' perceptions of Voluntariness of PSP Use increase, their perceived choice to use PSP will increase.

H10: As Software Developers' perceptions of Voluntariness of PSP Use increase, their perceived process control over PSP will increase.

H11: As Software Developers' perceptions of PSP Training increase, their perceived predictability with PSP will increase.

H12: As Software Developers' Degree of Novelty with PSP increases, their perceived predictability with PSP will decrease.

Table 25 shows the results of testing the hypotheses relating Developer Involvement and IT Environment variables to Perceived Control variables. Developer Involvement in the adoption of PSP has a significant, positive association with the developer's perception of choice in PSP use, as shown by the support of H7. Thus, it can be established that increases in developer involvement in such areas as setting objectives of the IT use, assessing the costs and benefits of the IT use, assessing alternative ways to meet development needs, etc. can have a substantial impact on the successful diffusion of the IT.

Dependent Variable	Independent Variable	Hypothesis	Hypothesized Direction	Beta	Sig.
Choice	Involvement	H7	+	**.274**	**.021**
	Champion	H8	-	.205	.178
	Voluntariness	H9	+	**.686**	**.000**
Process	Voluntariness	H10	+	**.622**	**.000**
Predictability	Training	H11	+	**.270**	**.081**
	Novelty	H12	-	.136	.372

Table 25: Results of Testing Involvement and Environmental Factors Hypotheses

Influences of the environmental factors on perceived control are mixed, with three of the five hypotheses supported. Hypothesis H9 relates increases in software developer perceived Voluntariness to increases in perceived Choice. Support for H9 indicates that voluntary use of PSP resulted in greater perceptions of control over software development work in terms of control over when to use PSP. This result also supports Moore and Benbasat's assertion that mandated use of an IT takes away freedom of choice from IT users [Moore 91]. Hypothesis H10 relates increases in software developer perceived voluntariness to increases in perceived

process control. Support for H10 gives added importance to enhancing perceptions of voluntary use as it shows that the freedom to decide whether or not to use a tool can positively influence one's perception of the freedom he has over how to best accomplish his work.

The effectiveness of training has also been found to be a significant factor in enhancing perceived control through its effect on enhancing predictability of work. Support for H11 suggests that high quality, thorough, and timely training on new software development techniques can give software developers more understanding of how they can expect to perform their software development tasks using the new software development technique.

Two environmental factors are found to not significantly impact perceived control: Champion Support and Degree of Novelty. Interestingly, these two factors are the only factors hypothesized to have a negative association with perceived control variables. Not only are the Betas insignificant (Prob>t = .178 for Champion Support, and .372 for Novelty), but their directions are the opposite of those hypothesized. The next section of this chapter, Post-Hoc Analyses, examines possible reasons for the non-support of these two hypotheses.

4.3.3 Post-Hoc Analyses

Five of the 12 study hypotheses could not be supported by the study data. Possible reasons for lack of support for one of these hypotheses (H3) were discussed in the previous section. The purpose of this section is to explore reasons for why the remaining hypotheses (H2, H4, H8 and H12) may not have been supported. We explore these reasons through post-hoc analyses of data.

Hypotheses H2 and H4 relate a software developer's perception of Choice in using PSP and his perception of the degree of behavioral (Process) control he has when using PSP to subsequent Use of PSP. Neither of these hypotheses was supported by study data. However, the IT Use variable as a measure of IT Success has been suggested to be inappropriate when Use of the IT is mandated or involuntary [DeLone 92, Melone 90]. Thus, we undertake an additional analysis to better isolate the potential relationships between perceived choice, process control, and PSP Use. A repeat of the regression model number 2 (see Table 21) is run with data representing those observations where PSP Use is voluntary. We split these data based on the subjective criteria that the respondents' mean score on the Voluntariness of Use variable is greater than 4.0. The resulting sample size for this analysis is 27. Figure 10 shows the results of this post-hoc regression.

Dependent Variable: USE

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Prob>F
Model	3	24.275	8.092	.821	.497
Residual	20	197.058	9.853		
Total	23	221.333			

Root MSE	3.1389	R-square	0.110
Dep Mean	7.1667	Adj R-sq	-0.024

Parameter Estimates

Variable	DF	Std. Beta Estimate	Std. Error	T for Ho: Parameter=0	Prob>T
PROCESS	1	.082	.350	.388	.702
CHOICE	1	.103	.547	- .490	.630
PREDICT	1	.293	.506	1.378	.183

Figure 10: Regression Results for Use-Related Hypotheses with Voluntary Users

In this model, none of the perceived control variables are found to have a significant relationship to PSP Use; in fact, the model itself is non-significant in explaining PSP Use (Prob>F = .497). The sample size used in this analysis is only 27 observations. At an alpha level of .05 and an effect size of .4, power analysis using tables in [Cohen 88] suggests that a sample size of at least 48 would have been needed to get to a .7 level of power. Thus, the applicability of the IT Use variable in mandated versus voluntary situations cannot adequately be tested in this study, and its impact in the results of testing hypotheses H2 and H4 is unknown. This impact should be pursued in a future study with a larger sample of voluntary users.

In Chapter 3, it was noted that Taylor and Todd found a positive, significant relationship between Perceived Behavioral (Process) Control and IT Use [Taylor 95]. However, their operationalization of behavioral control involved access to resources needed to use the IT while this study operationalized this construct as freedom from management and/or IT constraints when using the IT. The contradictory results between the present study and previous research suggest that further refinement of the Perceived Behavioral (Process) control variable may be needed. Also, as this is the first study that directly examines the relationship between perceived degree of Choice and IT Use, these relationships should be further examined in future studies with a larger sample of voluntary users in order to further validate the findings in this study.

Hypothesis H8 relates increases in the degree of Champion Support for an IT innovation to decreases in a software developer's perception of choice in using the innovation. Data analysis results presented in the previous section do not support this hypothesis. The regression equation used to test hypotheses H7, H8, and H9 was as follows:

$$CHOICE = B_0 + B_1INVOLVE + B_2CHAMPION + B_3VOLUN$$

In the Model and Hypotheses Testing Strategy section of this thesis, it was noted that a high degree of negative correlation existed between the Champion Support variable (CHAMPION) and the Voluntariness of Use variable (VOLUN, r = -0.596, p=.001). Thus, as a post-hoc analysis, the above regression equation was split into two separate equations. Two regressions were run to determine if the VOLUN and CHAMPION variables are significant predictors of perceived choice when each appears in the regression equation without the other variable. The results of these additional regressions appear in Table 26. The results show that each variable does, in fact, significantly relate to CHOICE when the effects of the other variable are not included in the equation. Further, Table 26 shows that the beta values for Voluntariness of Use (VOLUN) and Developer Involvement (INVOLVE) do not dramatically change whether CHAMPION appears in the regression equation or not. Therefore, we can conclude that both hypotheses H8 and H9 are supported.

Dependent Variable	**Independent Variable**	**Hypothesis**	**Hypothesized Direction**	**Beta (both variables in equation)**	**Beta (variables in separate equations)**
Choice	Involvement	H7	+	**.274 (p=.021)**	**.323 (p=.024)** (w/CHAMPION) **.266 (p=.027)** (w/VOLUN)
	Champion	H8	-	.205 (p=.178)	**-.236 (p=.094)**
	Voluntariness	H9	+	**.686 (p=.000)**	**.553 (p=.000)**

Table 26: Post-Hoc Examination of Hypothesis H8

Hypothesis H12 relates increases in a software developer's perception of novelty of the IT innovation to decreases in the developer's ability to predict his work outcomes when using the innovation. Data analysis results presented in the previous section do not support this hypothesis. The regression equation used to test hypotheses H11 and H12 was as follows:

$$PREDICT = B_0 + B_1NOVELTY + B_2TRAINING$$

One possible explanation for the lack of effect of novelty on predictability could be that the level of training received by the developer is of high enough quality as to offset the potential negative effect of the newness of the innovation. In order to test this explanation, an additional regression is run that includes an interaction term representing the interaction between NOVELTY and TRAINING. Figure 11 shows the result of this additional regression. The in-

teraction between Novelty and Training (TRNXNOV) was insignificant (Beta = -1.168, Prob>t = .305), suggesting that this explanation for the results does not hold.

Dependent Variable: PREDICT

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Prob>F
Model	3	10.860	3.620	1.770	.169
Residual	39	79.745	2.045		
Total	42	90.605			

Root MSE	1.4299	R-square	0.120
Dep Mean	4.0581	Adj R-sq	0.052

Parameter Estimates

Variable	DF	Std. Beta Estimate	Std. Error	T for Ho: Parameter=0	Prob>T
NOVELTY	1	1.177	1.175	1.163	.252
TRAINING	1	.721	.805	1.570	.124
TRNXNOV	1	-1.168	.202	-1.040	.305

Figure 11: Regression Results for NOVELTY and TRAINING Interaction

Another possible explanation for the lack of effect of novelty on predictability may be that despite the newness of the innovation to the developer, his IS experience in general may give him a sufficient mental model of what to expect from the innovation so that his ability to predict work outcomes using the innovation is not affected. Data on the number of years of IS experience the developer has (ISYEARS) is included in the General Information section of the survey instrument. To test this possible explanation, an additional regression is run that includes an interaction term representing the interaction between NOVELTY and ISYEARS. Figure 12 shows the results of this additional regression run. The lack of significance for the NOVXISYR interaction term at an alpha level of .1 suggests that the foregoing explanation is not supported by study data.

Dependent Variable: PREDICT

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Prob>F
Model	4	11.579	2.895	1.392	.255
Residual	38	79.025	2.080		
Total	42	90.605			

Root MSE	1.4421	R-square	0.128
Dep Mean	4.0581	Adj R-sq	0.036

Parameter Estimates

Variable	DF	Std. Beta Estimate	Std. Error	T for Ho: Parameter=0	Prob>T
NOVELTY	1	.154	.359	.497	.622
TRAINING	1	.289	.267	1.891	.066
NOVXISYR	1	-.026	.038	- .044	.965
ISYRS	1	- .159	.165	- .304	.763

Figure 12: Regression Results for NOVELTY and ISYEARS Interaction

A final possible explanation for the lack of effect of novelty on predictability may simply be that the measurement tool for the Novelty variable contains too much random error to be reliable. As noted in the Reliability section of Chapter 3, the scale developed to measure degree of Novelty has a Cronbach alpha level of 0.73, the lowest of the 10 scales developed for this study. Thus, future research can focus on developing a more reliable measure of Degree of Novelty of an IT innovation, and then re-address the analysis of hypothesis H12.

4.3.4 Summary of Results

This chapter presents results of the study data analyses and discusses the results in terms of their support (or lack of support) for the research model. One of the key findings of these analyses is that a software developer's perception of control over his work when using an innovative software development technique is significantly related to the satisfaction of that developer with use of the technique. Support of H1 and H5, as well as the significance of the relationship hypothesized in H3, promote this finding. Thus, Perceived Control is established as an important factor in determining a software developer's Satisfaction with a software development innovation.

Another significant finding is that there are manager controlled factors that can enhance a software developer's perception of control, thereby increasing the chances that the innovative technique will be diffused into software development practice. These factors include the software developer's involvement in the development technique adoption process (H7 support), the degree of champion support for the IT (H8 support), the degree to which Use of the development technique is of free volition (H9 and H10 support), and the availability and adequacy of training on the new development technique (H11 support). Table 27 summarizes the findings for the individual research hypotheses.

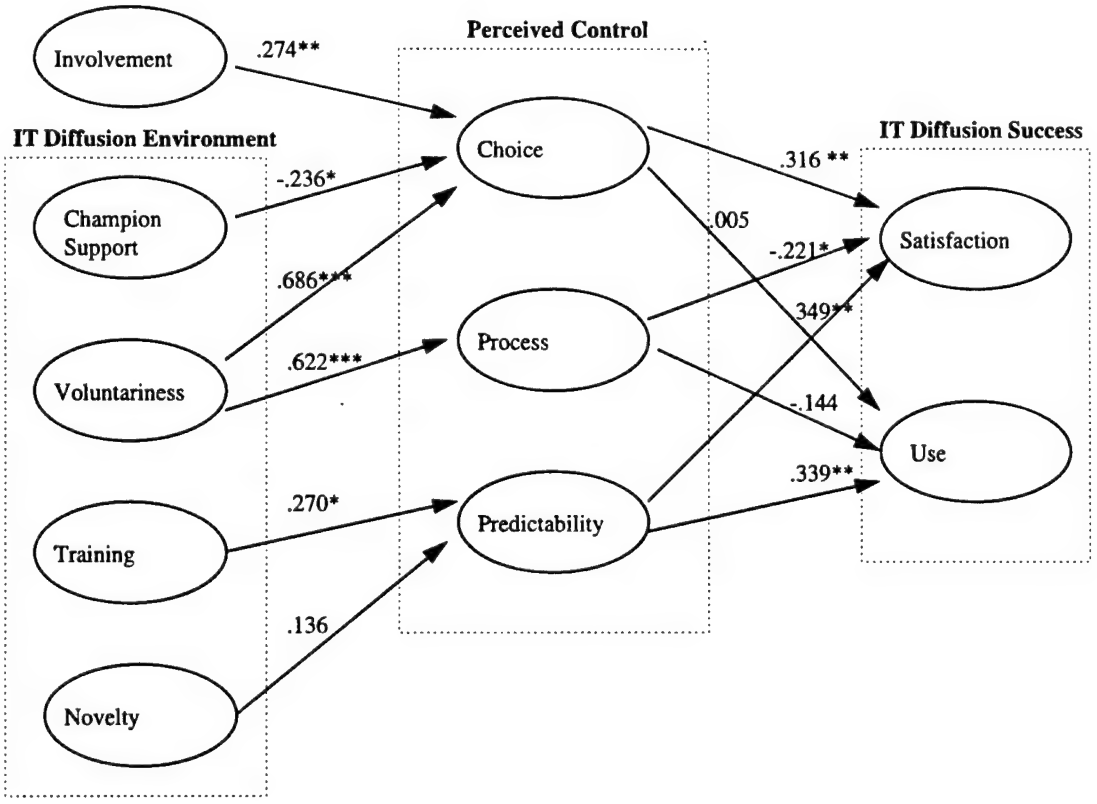
Hypothesis	Dependent Variable	Independent Variable	Supported?
H1	Satisfaction	Choice	Yes
H3		Process	No*
H5		Predictability	Yes
H2	Use	Choice	No
H4		Process	No
H6		Predictability	Yes
H7	Choice	Involvement	Yes
H8		Champion	Yes
H9		Voluntariness	Yes
H10	Process	Voluntariness	Yes
H11	Predictability	Training	Yes
H12		Novelty	No

(= relationship is significant, but in opposite direction of hypothesis)*

Table 27: Summary of Hypothesis Support

4.3.5 Mediating Effects

Figure 13 presents the research model with the path coefficients reported previously in Tables 24 through 26. The model implies that Developer Involvement and Environmental variables indirectly impact IT Diffusion Success through their direct impacts on Perceived Control variables. In order to check this assertion, two additional regressions are run to determine if there exist significant, direct paths between Developer Involvement and Environmental variables, and IT Diffusion Success variables. Tables 28 and 29 show the results of these regressions. There are two Beta values reported for most of the independent variables. This is due to the multicollinearity that exists between the Champion Support variable and the Voluntariness of Use variable that requires that two separate regression models be used to assess the effects of the independent variables on Use, one that includes Champion Support and excludes Voluntariness, and the other that includes Voluntariness and excludes Champion Support. This splitting of the regression runs is done to prevent spurious results in Beta coefficients that may be found when two highly correlated independent variables are present in the same regression equation.



(= significant at .1; ** = significant at .05; *** = significant at .01)*

Figure 13: Path Coefficients for Research Model

Independent Variable	Beta (all variables except Voluntariness) Prob>F = .027	Sig.	Beta (all variables except Champion Support) Prob>F = .001	Sig.
Developer Involvement	**.260**	**.073**	**.292**	**.029**
Champion Support	.193	.175	n/a	n/a
Voluntariness	n/a	n/a	-.414	**.003**
Training	**.296**	**.036**	**.383**	**.005**
Novelty	.150	.285	.099	.446

Table 28: Results of Testing for Direct Effects of IVs on PSP Use

The results shown in Table 28 suggest that there are direct impacts of Developer Involvement, Voluntariness, and Training on PSP Use. While these results are consistent with those of Iivari, they contradict the implication of the present research model, which is that the effects of Developer Involvement and IT Diffusion Environment variables on PSP Use are solely through their impacts on the developers' perception of control [Iivari 96].

The results shown in Table 29 similarly suggest that there is a direct effect of IT Diffusion Environment on Satisfaction with PSP. Specifically, there is a positive and significant relationship between Training and Satisfaction with PSP (Betas= .294 and .320, p=.045 and .038). Tables 30 and 31 show the total effects of each of the model independent variables on the two IT Diffusion Success variables. The total effect is calculated as the sum of the direct effect of the independent variable on the dependent variable, and the indirect effect of the independent variable through its impact on Perceived Control. Indirect effect is calculated by multiplying contributing path coefficients. In those cases where two path coefficients were calculated (for the multicollinearity problem), the path coefficients are averaged. Where paths are non-significant, they are assumed to be zero for purposes of calculating total effects. The data in Table 30 shows that for ensuring Satisfaction with PSP, Training and Perceived Control are important factors. Table 31 shows that Involvement, Voluntariness (actually mandated use since the Beta for Voluntariness is negative), and Training are key to ensuring PSP Use, as is the Predictability dimension of Perceived Control.

Independent Variable	Beta (all variables except Voluntariness) Prob>F = .119	Sig.	Beta (all variables except Champion Support) Prob>F = .250	Sig.
Developer Involvement	.149	.319	.117	.439
Champion Support	.235	.114	n/a	n/a
Voluntariness	n/a	n/a	-.120	.437
Training	**.294**	**.045**	**.320**	**.038**
Novelty	-.126	.388	-.124	.409

Table 29: Results of Testing for Direct Effects of IVs on Satisfaction with PSP

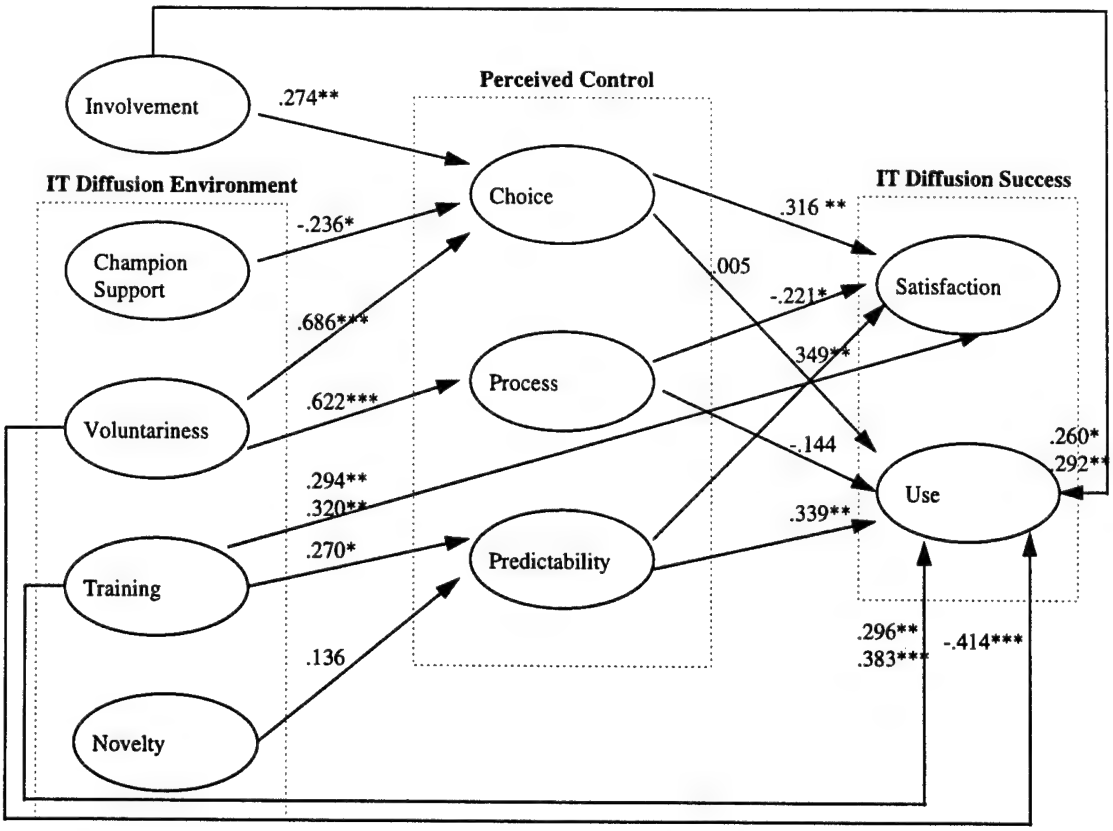
Variable	Direct Effect	Indirect Effect	Total Effect
Developer Involvement	-	.086	.086
Champion Support	-	-.075	-.075
Voluntariness	-	.079	.079
Training	.307	.095	.402
Novelty	-	-	-
Choice	.316	-	.316
Process Control	-.221	-	-.221
Predictability	.349	-	.349

Table 30: Total Effects of Variables on Satisfaction with PSP

Variable	Direct Effect	Indirect Effect	Total Effect
Developer Involvement	.276	-	.276
Champion Support	-	-	-
Voluntariness	-.414	-	-.414
Training	.339	.092	.431
Novelty	-	-	-
Choice	-	-	-
Process Control	-	-	-
Predictability	.339	-	.339

Table 31: Total Effects of Variables on PSP Use

From the foregoing analyses, we can conclude that the mediating variables proposed in the research model are not the sole vehicles through which Developer Involvement and IT Diffusion Environment impact IT Diffusion Success. The current research model is therefore modified to show that there are direct effects of Developer Involvement and IT Diffusion Environment on IT Diffusion Success. The modified research model, including path coefficients, is shown in Figure 14. The modified conceptual research model is shown in Figure 15.



(= significant at .1; ** = significant at .05; *** = significant at .01)*

Figure 14: Modified Path Coefficient Model

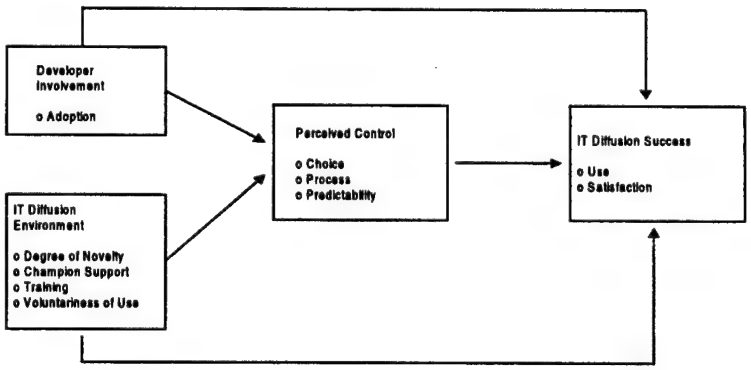


Figure 15: Modified Research Model

In summary, we have found in this research that there are four factors that are important in determining the successful diffusion of innovative software development techniques. Developer Involvement in the adoption process of the technique is important as it directly influences the developer's subsequent Satisfaction with using the technique. Timely and quality Training is also important to the success of the technique as it directly influences both the developer's Satisfaction with the technique as well as his sustained Use of the technique. Management mandates (i.e., Voluntariness) and Champion push for the use of the development technique have also been found to have implications on a developer's Satisfaction with using the technique because of their influences on the developers' perceived control over his work when using the technique. The next chapter discusses the implications of these findings to academia, to the SEI, and to the software engineering community at large. We also present opportunities for future research.

5 Conclusions

5.1 Introduction

This dissertation research deals with the diffusion of innovative software development techniques (referred to as *IT*). Specifically, we have investigated how (1) software Developer Involvement in the IT adoption process and (2) characteristics of the IT's Diffusion Environment contribute to the successful Diffusion of the IT. The goal of this research is to improve the efforts to diffuse new and effective software development techniques in development organizations.

Factors affecting the successful diffusion of software development techniques are identified and described in Chapter 2. A research framework and subsequent model are developed for the Diffusion of Software Development Techniques (Chapters 2 and 3), and are used to guide the subsequent empirical study. A field study was undertaken to test the relationships proposed in the research model. The results of this study (described in Chapter 4) confirm that Developer Involvement and certain IT Diffusion Environment characteristics are key to the successful diffusion of software development techniques. Developer Involvement, Voluntariness, and Training directly affect IT diffusion. These variables as well as Champion Support also indirectly affect IT diffusion through their impacts on software developers' perception of personal control. This chapter discusses the contributions, implications, and limitations of this research, as well as opportunities for future research in the diffusion of software development techniques.

5.2 Contributions

This research makes several important contributions to the research community as well as the software engineering practitioner community. These contributions and their implications are discussed in the following sections.

5.2.1 Research

There are three primary contributions of this research to the research community:

- an extension of Diffusion of Innovations theory
- the development of an integrated model of IT Diffusion
- the development of a comprehensive measure for perceived control

Diffusion of Innovations (DOI) theory is enhanced by the inclusion of Developer Involvement in the adoption process as an individual/environmental factor impacting IT Diffusion,

and the inclusion of Perceived Control as an attitude that impacts successful IT Diffusion. While the IS implementation literature abounds with studies of end-user involvement in the software development process and its impact on user satisfaction and IT use, virtually no prior DOI research has investigated the involvement construct for its potential relationship to IT Diffusion Success. Iivari investigated this direct relationship, but used an organizational level of analysis and used an objective measure of participation (e.g., number of developers involved in the selecting, planning, and implementation of CASE tools) [Iivari 95]. He subsequently found no significant relationship between developer involvement and CASE use. The present research extends DOI theory by demonstrating that Developer Involvement in the adoption process is significantly related to the developer's subsequent Use of the IT.

Previous DOI theory identifies Champion Support, Voluntariness, and Training as important to the diffusion of IT; however, as with previous DOI studies, the present study provides mixed support of the importance of these factors in directly impacting IT Use. Training and Voluntariness are found to be significantly related to IT Use in the current study; Champion Support is found to have a non-significant direct relationship to IT Use. However, by extending DOI theory to include social psychology theory on Perceived Control, the importance of all three of the aforementioned factors becomes even greater. The present research confirms the importance of software developers' perception of control over their work in determining the extent of an IT's diffusion. Perceived Control is found in this study to be strongly related to Developer Satisfaction and moderately related to IT Use. Further, Perceived Control is found in this study to be strongly influenced by Champion Support, Voluntariness, and Training. Thus, this study provides greater support for the importance of these previously identified factors in determining IT Diffusion Success because they directly impact IT Diffusion Success, and because they influence an important attitude (Perceived Control) which is also directly related to IT Diffusion Success.

Based on the above discussion, it can be seen that a second major contribution of this research to the research community is the development of an integrated model for IT diffusion success that provides greater explanatory power in understanding IT Diffusion. Relevant theories and ideas from four research streams (software engineering, IS implementation, diffusion of innovations, and social psychology) are integrated to identify and provide a greater understanding of factors that impact successful diffusion of software development techniques.

The third key contribution to IS research is the development of a comprehensive measure of perceived personal control over work in an IS context. The development of such a measure addresses concerns of previous IS researchers in this area [Baronas 88]. The perceived control instrument is more comprehensive than existing measures used in IS research because it incorporates three dimensions of perceived control (Perceived Choice, Predictability, and Behavioral Control), as opposed to existing measures in IS research, which incorporate Behavioral Control only [Taylor 95, Mathieson 91]. The construct of Perceived Control over one's work has been found to be directly related to IT Diffusion Success. As such, it is important to

understand how to measure developers' perceived level of control in order to better predict how a software development technique may fare in an organization. This research addresses this issue by the development of a reliable and comprehensive measure of perceived control.

5.2.2 Practice

The current research contributes to the practitioner community in three key ways:

1. identifying additional indicators of IT diffusion success
2. identifying factors that impact IT diffusion success
3. demonstrating the importance of a software developer's perceived control over his work

There are a number of different categories of practitioners who will be interested in these results. Industrial change agents and champions of new IT innovations will find support and guidance for successfully introducing new ideas and technologies into practice. Software development managers and the software developers themselves will take away a better understanding of how to integrate new technologies into their projects and make them more effective.

The framework developed in this research identifies non-technical indicators of IT diffusion success such as Developer Satisfaction with the IT as well as subsequent Use of the IT. While traditional software engineering literature focuses more on "objective" indicators of IT success such as error reduction and productivity improvements, these improvements in software development will not be realized without the incorporation and sustained use of the software development technique into practice. Thus, the research framework developed in this study includes behavioral indicators of success, as they are clearly relevant to the diffusion of software development techniques.

A second key contribution of this research to practice is the identification of factors that impact IT diffusion success. Traditional software engineering literature has focused on identifying and developing the tools and techniques that improve software quality and programmer productivity. However, little attention is given to how to best introduce the IT into an organization to ensure its successful and sustained use. The framework developed in this research identifies factors that influence the diffusion of software development techniques in practice. These factors can be influenced and controlled by IS management in order to better ensure successful diffusion of innovative software development techniques.

One of the central constructs explored in this research is the software developer's perception of personal control over his software development work when using a software development technique. Results of data analysis support the research framework, which shows that there is a direct relationship between a software developer's perception of control and his Satisfaction with using the software development technique. Specifically, this research has demonstrated that developers are more satisfied with using software development innovations (1) when they have a choice in using that innovation (H1), (2) when the innovation increases the pre-

dictability of their work (H5), and (3) when the innovation provides more controls in their work (H3). This research has also demonstrated that developers tend to subsequently Use a software development technique when that technique provides more predictability in their software development work (H6). *The first finding implies that managers should provide more freedom to individual developers to decide when to use innovative software development techniques and how best to adapt the technique to their task. The latter two findings provide support for the use of more formal, structured engineering practices in software development as a way to provide more controls in software development, and subsequently, to provide greater Satisfaction with and sustained Use of software development techniques.*

This research goes further to identify factors that influence developers' perceived control and therefore influence diffusion success. Results of data analyses show that greater levels of developer involvement in the process of adopting a software development innovation are associated with enhanced feelings of perceived control (H7) as well as enhanced satisfaction with the IT innovation. *Thus, software development managers are encouraged to involve software developers in adoption tasks such as identifying areas for improvement in software development, identifying tools and techniques on the market that may improve these software development areas, assessing costs and benefits of using the tool or technique, and determining objectives of using new tools and techniques.*

Data analysis results also show that training on the software development innovation is key to determining its success in a development organization (H11). *Managers should be careful to provide comprehensive and high-quality training to software developers in order for developers to effectively use the IT in their development work.* Managers should also provide sufficient time and effective timing for training; in other words, training should be received just prior to expected use of the software development technique, and sufficient time should be allowed for developers to grasp the tool/technique concepts.

The degree to which use of the software development technique is Voluntary has also been found to have a direct relationship to the software developer's perception of control. Greater degrees of Voluntary use are associated with higher levels of perceived control (H9 and H10). *This finding suggests that where possible, IS managers should perhaps encourage, rather than mandate, use of the software development technique, thereby creating a perception of voluntary use.*

Finally, data analysis reveals that greater degrees of Champion Support for an IT are associated with lowered levels of perceived control (H8). While traditional IS research has strongly advocated champion support for IT in organizations, the finding in this study suggests that too much push from an IT champion may adversely impact software developers' perceived control, potentially decreasing the chance for successful diffusion of the IT in the organization [Alexander 89, Beath 91]. This statement is not meant to discourage champion support for IT innovations. While no significant, direct relationship between Champion Support and IT Diffusion Success was found in this research, previous IS studies have found that in-

creases in Champion Support (or more specifically, Management Support) are directly associated with increases in IT Use [Iivari 96]. *What the present research does suggest, however, is that while potentially effective in increasing IS Use, increases in Champion Support should be augmented with other mechanisms, such as adequate training and increased levels of developer involvement, so that developers maintain a sense of personal control in their work.*

5.2.3 Software Engineering Institute (SEI)

One of the key contributions of this research to the SEI is guidance on effective planning for and positioning of innovative software development practices. While there have been several studies that examine the diffusion of software development *tool* innovations such as CASE tools and object-oriented languages, there have been no studies that examine the diffusion of innovative software development processes, such as the PSP approach [Iivari 96, Fichman 97, Chau 96]. By expanding the concept of "IT Innovation" to include software development processes and by using a process such as PSP to test the validity of our research framework, the framework can be generalized to a broader category of software development techniques. Therefore, as the SEI develops IT innovations such as standards, procedures, and tools to improve software development effectiveness, the results of this research can be used by the SEI to ensure organizations are prepared to provide an environment that facilitates the success of the IT.

Another contribution of this research to the SEI is the collection of data from the software engineering community on Satisfaction with PSP and Use of PSP. Much data has been collected about the PSP approach and there is much interest at SEI and in the software engineering community in general in the results of this study. Thus, a key contribution of this study is to give the SEI, as well as the software engineering community in general, feedback on how successful PSP has been in meeting its objective of transforming software developers into software engineers. This feedback includes:

- descriptive information on the types of projects in which the PSP approach is being used
- the primary phases of development in which PSP is being used
- the effectiveness of PSP training
- impacts of PSP on predictability of work
- issues and challenges involving Use of PSP in practice (via the Comments section of survey)

Further, the study gives support to the assertion of the author of the PSP approach, which states that PSP is designed to give software engineers more control over their software development work [Humphrey 95]. Indeed, the average software engineer surveyed reported generally positive feelings of personal control when using PSP (see Descriptive Statistics section).

Of further interest to the SEI is the low number of PSP users and participants in this study. As noted in the Procedures for Survey Administration section of this dissertation, the SEI con-

tacted all known individuals who have conducted PSP training or have been trained in PSP. Approximately 90 individuals were identified and contacted by the SEI and the authors to participate in this study. Participation in the study involved the completion of the study questionnaire by the individual, or distribution of the questionnaire by the individual to those developers in his or her organization who have used PSP. Approximately 30 of these individuals indicated that they would be willing to participate in the study. The remaining contacts either did not respond or indicated that they could not participate because they did not use PSP beyond initial training. Some of the reasons given (either verbally or via survey comments) for non-use of PSP are that the nature of the developer's work has changed since PSP training, that there was too much work involved in applying PSP, and that there had not been enough time for the organization to adapt PSP to its environment.

A total of 131 surveys were distributed to the 31 individuals who agreed to participate in the study. Of these 131 surveys, only 56 surveys were completed and returned, resulting in a response rate of 42%. While this response rate is "better than average" in field research, it is lower than expected from the SEI. Future research should be directed toward a better understanding of the inhibitors to PSP use.

5.3 Limitations

Every effort has been made in this research to develop a comprehensive research framework, develop reliable and valid measures of study variables, and analyze the study data using robust and powerful statistical techniques. Further, a research design was chosen that maximizes generalizability of research findings to the software development community. However, limitations of this research do exist and are discussed next.

5.3.1 Sample Size and Statistical Technique

Perhaps the most obvious limitation of this research is the low sample size, which resulted in the use of a series of multiple regressions in order to test the research model. The sample size of this study provided adequate power for the testing of research hypotheses (70–85%). However, because the research model represents a structural model with both latent and observed variables, a more powerful statistical technique such as structural equation modeling would have been a more desirable way to test the adequacy of the research model to describe the sample data. Thus, a recommendation for future research is to repeat this study with a larger sample size (a minimum of 100 has been suggested by Hayduk , although most recommendations suggest 200–400 respondents) [Hayduk 87].

5.3.2 Retrospective Data Collection

Six of the ten scales developed for this study require respondents to recall previous experiences in order to answer the questions. However, responses based on recollections have the potential for bias in that more recently encountered experiences may overshadow or skew judgments about less recent experiences [Hufnagel 94]. However, because this research is interested in software development innovations no more than two years old, and because the

average experience with PSP in this study is 12 months, this potentially biasing effect is minimized.

5.3.3 Causality

While the field survey design used in this research is appropriate for the investigation of hypothesized relationships, this type of research prevents conclusions regarding causality. The use of a randomized or quasi-experimental design where levels of the research variables are manipulated would address this limitation. However, as stated in the Research Design section of this thesis, the relatively large number of independent variables that would have to be manipulated combined with the amount of time and resources required to manipulate some of these variables makes such a design more difficult to implement.

5.3.4 Additional Model Variables

The research framework for Software Development Technique Diffusion developed in this thesis integrates ideas and theories from four research areas. Thus, this framework is robust and provides increased understanding of the diffusion process. However, there are variables, which are not included in the present framework, that may provide greater explanatory power as well as greater understanding of some of the non-supported research hypotheses. Chapter 4 discusses one such additional variable that should be considered in future enhancements to the current model: Desired Control. This variable may possibly interact with perceived control to affect the impacts on developer satisfaction.

Survey comments provide input for other possible enhancements to the current research framework. One survey respondent noted that "being methodic, rational, objective, [and] having a sense of organization...are characteristics that I discovered in those that I know who continued using PSP." Indeed it may be that certain personality types tend to adopt structured software development innovations more so than others. Such a personality type variable could be considered in future enhancements of the research framework.

Similarly, a few survey comments reflect the fact that developer motivation is a key individual attribute that determines the success of an IT innovation. One survey respondent, in explaining why PSP was not used on a "real" project, noted that "...group procedures don't do anything to encourage individuals to use PSP and I haven't been able to motivate myself to use PSP on my own."

5.4 Future Research

The previous section on Limitations also includes suggestions for future research on the diffusion of software development techniques. These recommendations included replicating the current study with a larger sample size, facilitating the use of a more powerful statistical technique such as structural equation modeling, and including additional variables in the research framework. Other suggestions for future research follow.

5.4.1 Additional Software Development Techniques

While this study uses PSP as an instance of an IT innovation in order to validate the research model, this study is not a PSP study. Because the research model is grounded in well-established theory, the model is expected to hold true for a wide range of software development innovations. Future research should validate this by using another software development technique innovation. Another advantage of repeating this study using a different IT innovation is that the IT that is chosen can be one which does not require implementation across all phases of the development cycle, giving the potential for greater usage in a two-year timeframe than PSP. And while the Quality and Productivity research model variables are not analyzed in this study, survey comments indicate that an IT that spans fewer development phases may provide more timely and accurate assessments of Quality and Productivity impacts than an IT such as PSP.

5.4.2 Novelty Measure

Data analysis finds no support of the hypothesized relationship between Degree of Novelty and Predictability (H12). However, as noted in Chapters 3 and 4, the Novelty scale has the lowest level of reliability of all 10 study scales (Cronbach's alpha = .73). This reliability is consistent with existing measures of Novelty [Chau 96]. Thus, another area for future research is in developing a more reliable measure of the degree of newness of an IT to the developer (Degree of Novelty). Once such a measure is developed, hypothesis H12 should be retested to see if there is, in fact, support for the relationship between Novelty and Predictability.

5.4.3 Remaining Framework Variables

Finally, the research model tested in this study was a subset of the research framework developed in this thesis. Data has been collected on most of the additional framework variables, including Application Domain, Perceived Ease of Use, Perceived Usefulness, Perceived Quality Impacts, and Perceived Productivity Impacts. However, these variables and their associated relationships are not analyzed in this report. A priority for future research is to begin analyzing these data in an effort to validate the entire research framework.

5.5 Concluding Remarks

This research has examined factors that impact the Diffusion of Software Development Techniques. The results of this study make many important contributions to research and practice, as described in previous sections of this chapter. However, the overall contribution of this research is improving the effectiveness of software development efforts by increasing the likelihood that beneficial software development innovations will be used.

References

[Adams 92] Adams, D., R. Nelson, and P. Todd. "Perceived Usefulness, Ease of Use and Usage of Information Technology: A Replication," *MIS Quarterly*, (16:2), June 1992, pp. 227-248.

[Ajzen 85] Ajzen, I. "From Intentions to Actions: A Theory of Planned Behavior," in *Action Control: From Cognition to Behavior*, J. Kuhl and J. Beckmann (Eds.), Springer Verlag, NY, 1985, pp. 11-39.

[Ajzen 91] Ajzen, I. "The Theory of Planned Behavior," *Organizational Behavior and Human Decision Processes*, (50), 1991, pp. 179-211.

[Alexander 89] Alexander, M. "The Adoption and Implementation of Computer Technology in Organizations: The Example of Database Machines," Doctoral Dissertation, School of Business, Indiana University, 1989.

[Bailey 83] Bailey, J. and S. Pearson. "Development of a Tool for Measuring and Analyzing Computer User Satisfaction," *Management Science*, (29:6), May 1983, pp. 519-529.

[Baker 75] Baker, T. "Structured Programming in a Production Programming Environment," *IEEE Transactions on Software Engineering*, June 1975, pp. 235-246.

[Bandura 83] Bandura, A. "Temporal Dynamics and Decomposition of Reciprocal Determinism: A Reply to Phillips and Orton," *Psychological Review*, (90), 1983, pp. 166-170.

[Baronas 88] Baronas, A. and M. Louis. "Restoring a Sense of Control During Implementation: How User Involvement Leads to System Acceptance," *MIS Quarterly*, (12:1), March 1988, pp. 111-124.

[Baroudi 86] Baroudi, J., M. Olson, and B. Ives. "An Empirical Study of the Impact of User Involvement on System Usage and Information Satisfaction," *Communications of the ACM*, (29:3), March 1986, pp. 232-238.

[Beath 91] Beath, C. "Supporting the Information Technology Champion," *MIS Quarterly*, (15:3), September 1991, pp. 355-372.

[Blalock 71] Blalock, H. *Causal Models in the Social Sciences*, Aldine-Atherton, 1971.

[Cavaye 95] Cavaye, A. "User Participation in System Development Revisited," *Information and Management*, (28), 1995, pp. 311-323.

[Chau 96] Chau, P. "An Empirical Investigation of Factors Affecting the Acceptance of CASE by Systems Developers," *Information and Management*, (30), 1996, pp. 269-280.

[Constantine 95] Constantine, L. *Constantine on Peopleware*, Yourdon Press, Prentice-Hall PTR, 1995.

[Corah 70] Corah, N. and J. Boffa. "Perceived Control, Self-Observation, and Response to Aversive Stimulation," *Journal of Personality and Social Psychology*," (16:1), 1970, pp. 1-4.

[Cronbach 51] Cronbach, L. "Coefficient Alpha and the Internal Structure of Tests," *Psychometrika*, (16:3), 1951, pp. 297-334.

[Davis 89a] Davis, F. "Perceived Usefulness, Perceived Ease of Use, and User acceptance of Information Technology," *MIS Quarterly*, (13:3), September 1989, pp. 319-339.

[Davis 89b] Davis, F., R. Bagozzi, and P. Warshaw. "User Acceptance of Computer Technology: A Comparison of Two Theoretical Models," *Management Science*, (35:8), August 1989, pp. 982-1003.

[Davis 93] Davis, S. and R. Bostrom. "Training End-Users: An Experimental Investigation of the Roles of Computer Interface and Training Methods," *MIS Quarterly*, (17:1), March 1993, pp. 61-85.

[DeBrabander 77] DeBrabander, B. and A. Edstrom. "Successful Information Systems Development Projects," *Management Science*, (24:2), 1977, pp. 191-199.

[DeLone 92] DeLone, W. and E. McLean. "Information Systems Success: The Quest for the Dependent Variable," *Information Systems Research*, (3:1), March 1992, pp. 60-95.

[DeMarco 87] DeMarco, T. and T. Lister. *Peopleware: Productive Projects and Teams*, Dorset House Publishing Co., 1987.

[Doll 88] Doll, W. and G. Torkzadeh. "The Measurement of End-User Computing Satisfaction," *MIS Quarterly*, (12:2), June 1988, pp. 259-274.

[Doll 89] Doll, W. and G. Torkzadeh. "A Discrepancy Model of End-User Computing Involvement," *Management Science*, (35:10), October 1989, pp. 1151-1171.

[Fayad 96] Fayad, M., W. Tsai, and M. Fulghum. "Transition to Object-Oriented Software Development," *Communications of the ACM*, (39:2), February 1996, pp. 109-121.

[Ferguson 97] Ferguson, P., W. Humphrey, S. Khajenoori, S. Macke, and A. Matvya. "Results of Applying the Personal Software Process," *Computer*, May 1997, pp. 24-31.

[Fichman 97] Fichman, R. and C. Kemerer. "The Assimilation of Software Process Innovations: An Organizational Learning Perspective," *Management Science*, (43:10), October 1997, pp. 1345-1363).

[Fishbein 75] Fishbein, M. and I. Ajzen. *Belief, Attitude, Intention, and Behavior: An Introduction to Theory and Research*, Addison-Wesley Inc., 1975.

[Fowler 94] Fowler, P. "The Challenge of Transferring Software and Information Technology," in *Business Process Re-Engineering: Information Systems Opportunities and Challenges*, G. Glasson et al. (Eds.), Elsevier Science B.V. (North-Holland), pp. 79-88, 1994.

[Fowler 93a] Fowler, P. and L. Levine. *A Conceptual Framework for Software Technology Transition* (CMU/SEI-93-TR-031, ADA275637). Software Engineering Institute, Carnegie Mellon University, 1993. Available WWW: <URL: http://www.sei.cmu.edu/publications/documents/93.reports/93.tr.031.html>.

[Fowler 93b] Fowler, P. and L. Levine. *Technology Transition Pull: A Case Study of Rate Monotonic Analysis, Part 2* (CMU/SEI-93-TR-030). Software Engineering Institute, Carnegie Mellon University, 1993. Available WWW <URL: http://www.sei.cmu.edu/publications/documents/93.reports/93.tr.030.html>.

[Gersick 90] Gersick, C. and J. Hackman. "Habitual Routines in Task-Performing

Groups," *Organizational Behavior and Human Decision Processes*, (47), 1990, pp. 65-97.

[Giacobbe 95] Giacobbe-Miller, J. "A Test of the Group Values and Control Models of Procedural Justice from the Competing Perspectives of Labor and Management," *Personnel Psychology*, (48:1), Spring 1995, pp. 115-142.

[Gibbs 94] Gibbs, W. "Software's Chronic Crisis," *Scientific American*, September 1994, pp. 86-95.

[Gibson 97] Gibson, R. "Applied Software Process Improvement," *Proceedings of the Americas Conference on Information Systems*, August 1997, pp. 596-598.

[Goodhue 95] Goodhue, D. and R. Thompson. "Task-Technology Fit and Individual Performance," *MIS Quarterly*, (19:2), June 1995, pp. 213-236.

[Green 98] Green, G. *Examining the Impacts of Developer Involvement and IT Diffusion Environment on the Diffusion of Software Development Technique.* Doctoral Dissertation, College of Business Administration, University of South Florida, August 1998.

[Green 95] Green, G. and R. Collins. "An Interactionist Approach to Complexity in Computer-Supported Knowledge Work," *Proceedings of the Inaugural Americas Conference on Information Systems*, Pittsburgh, PA, 1995, pp. 83-85.

[Greenberger 86] Greenberger, D. and S. Strasser. "Development and Application of a Model of Personal Control in Organizations," *Academy of Management Review*, (11:1), 1986, pp. 164-177.

[Hartwick 94] Hartwick, J. and H. Barki. "Explaining the Role of User Participation in Information System Use," *Management Science*, (40:4), April 1994, pp. 440-465.

[Hayduk 87] Hayduk, L. *Structural Equation Modeling with LISREL: Essentials and Advances*. The John Hopkins University Press, 1987.

[Henderson 92] Henderson, J. and S. Lee. "Managing I/S Design Teams: A Control Theories Perspective," *Management Science*, (38:6), June 1992, pp.

757-776.

[Hevner 95] Hevner, A. and H. Mills. "Box-structured Requirements Determination Methods," *Decision Support Systems*, (13:3,4), March 1995, pp. 223-239.

[Hoffer 92] Hoffer, J. and M. Alexander. "The Diffusion of Database Machines," *Database*, (23:2), 1992, pp. 13-19.

[Holloway 96] Holloway, C. and R. Butler, "Impediments to Industrial Use of Formal Methods," *IEEE Computer*, (29:4), April 1996, pp. 25-26.

[Hufnagel 94] Hufnagel, E. and C. Conca. "User Response Data: The Potential for Errors and Biases," *Information Systems Research*, (5:1), March 1994, pp. 48-73.

[Humphrey 95] Humphrey, W. *A Discipline for Software Engineering*, Addision-Wesley, 1995.

[Humphrey 96] Humphrey, W. "Using a Defined and Measured Personal Software Process," *IEEE Software*, 1996, pp. 77-88.

[Humphrey 97] Humphrey, W. *Introduction to the Personal Software Process*, Addison-Wesley, 1997.

[Iivari 93] Iivari, J. "From a Macro Innovation Theory of IS Diffusion to a Micro Innovation Theory of IS Adoption: An Application to CASE Adoption," in Avison, D., Kendall J., and DeGross, J. (eds.), *Human, Organizational, and Social Dimensions of Information Systems Development*, North-Holland, Amsterdam, The Netherlands, 1993, pp. 295-320.

[Iivari 96] Iivari, J. "Why Are CASE Tools Not Used?" *Communications of the ACM* (39:10), October 1996, pp. 94-103.

[Ives 84] Ives, B. and M. Olson. "User Involvement and MIS Success: A Review of Research," *Management Science*, (30:5), May 1984, pp. 586-603.

[Ives 83] Ives, B., M. Olson, and J. Baroudi. "The Measurement of User Information Satisfaction," *Communications of the ACM*, (26:10), October 1983, pp. 785-793.

[Jackson 97] Jackson, C., S. Chow, and R. Leitch. "Toward an Understanding of the Behavioral Intention to Use an Information System," *Decision Sciences*, (28:2), Spring 1997, pp. 357-389.

[Kirsch 96] Kirsch, L. "The Management of Complex Tasks in Organizations: Controlling the Systems Development Process," *Organization Science*, (7:1), January-February 1996, pp. 1-21.

[Kwon 87] Kwon, T. and R. Zmud. "Unifying the Fragmented Models of Information Systems Implementation," in *Critical Issues in Information Systems Implementation*, R.J. Boland and R. A. Hirschheim (Eds.), John Wiley and Sons, Chichester, 1987, pp. 227-251.

[Langer 83] Langer, E. *The Psychology of Control*, Sage Publications, 1983.

[Linger 94] Linger, R. "Cleanroom Process Model," *IEEE Software*, (11:2), March 1994, pp. 50-58.

[Leonard-Barton 88] Leonard-Barton, D. and I. Deschamps. "Managerial Influence in the Implementation of New Technology," *Management Science*, (34:10), 1988, pp. 1252-1265.

[Locke 79] Locke, E. and D. Schweiger. "Participation in Decision-Making: One More Look," *Research in Organizational Behavior*, (1), 1979, pp. 265-339.

[Luqi 97] Luqi and J. Goguen. "Formal Methods: Promises and Problems," *IEEE Software*, (14:1), January 1997, pp. 73-85.

[Mantei 89] Mantei, M. and T. Teorey. "Incorporating Behavioral Techniques Into the Systems Development Life Cycle," *MIS Quarterly*, (13:3), September 1989, pp. 257-273.

[Mathieson 91] Mathieson, K. "Predicting User Intentions: Comparing the Technology Acceptance Model with the Theory of Planned Behavior," *Information Systems Research*, (2:3), September 1991, pp. 173-191.

[Melone 90] Melone, N. "A Theoretical Assessment of the User-Satisfaction Construct in Information Systems Research," *Management Science*, (36:1), January 1990, pp. 76-91.

[Monty 73] Monty, R., M. Rosenberger, and L. Perlmuter. "Amount and Locus of Choice as Sources of Motivation in Paired-Associate Learning,"

Journal of Experimental Psychology, (97), 1973, pp. 16-21.

[Moore 91] Moore, G. and I. Benbasat. "Development of an Instrument to Measure the Perceptions of Adopting an Information Technology Innovation," *Information Systems Research*, (2:3), 1991, pp. 192-222.

[Neumann 96] Neumann, P. "Using Formal Methods to Reduce Risks," *Communications of the ACM*, (39:7), July 1996, pg. 114.

[Nunnally 78] Nunnally, J. Psychometric Theory, New York: McGraw-Hill, 1978.

[Olson 81] Olson, M. and B. Ives. "User Involvement in System Design: An Empirical Test of Alternative Approaches," *Information and Management*, (4:4), 1981, pp. 183-196.

[Orlikowski 93] Orlikowski, W. "CASE Tools as Organizational Change: Investigating Incremental and Radical Changes in Systems Development," *MIS Quarterly*, (17:3), September 1993, pp. 309-340.

[Parnas 85] Parnas, D., P. Clements, and D. Weiss. "The Modular Structure of Complex Systems," *IEEE Transactions on Software Engineering*, (SE-11:3), March 1985, pp. 259-266.

[Pedhazur 82] Pedhazur, E. *Multiple Regression in Behavioral Research*, Holt, Rinehart, and Winston, 1982.

[Perlmuter 71] Perlmuter, L., R. Monty, and G. Kimble. "The Effects of Choice on Paired-Associate Learning," *Journal of Experimental Psychology*, (91), 1971, pp. 47-53.

[Pfleeger 97] Pfleeger, S. and L. Hatton. "Investigating the Influence of Formal Methods," *IEEE Computer*, February 1997, pp. 33-43.

[Poore 93] Poore, J., H. Mills, and D. Mutchler. "Planning and Certifying Software System Reliability," *IEEE Software*, (10:1), January 1993, pp. 88-99.

[Popper 59] Popper, K. *The Logic of Scientific Discovery*, New York: Basic Books, 1959.

[Powers 73] Powers R. and G. Dickson. "MIS Project Management: Myths, Opinions, and Reality," *California Management Review*, (15:3), 1973, pp. 147-156.

[Prescott 95] Prescott, M. and S. Conger. "Information Technology Innovations: A Classification by IT Locus of Impact and Research Approach," *Data Base*, (26:2-3), May/August 1995, pp. 20-41.

[Rai 94] Rai, A. and G. Howard. "Propagating CASE Usage for Software Development: An Empirical Investigation of Key Organizational Correlates," *OMEGA: The International Journal of Management Science*, (22:2), 1994, pp. 133-147.

[Rogers 82] Rogers, E. "Information Exchange and Technological Innovation," in *The Transfer and Utilization of Technical Knowledge*, D. Sahal, ed. Lexington, Mass., Lexington Books, 1982, pp. 105-123.

[Rogers 83] Rogers, E. *Diffusion of Innovations*, Third Edition, New York: Free Press, 1983.

[Rogers 95] Rogers, E. *Diffusion of Innovations*, Fourth Edition, New York: Free Press, 1995.

[Saiedian 96] Saiedian, H., J. Bowen, R. Butler, D. Dill, R. Glass, D. Gries, A. Hall, M. Hinchey, C. Holloway, D. Jackson, C. Jones, M. Lutz, D. Parnas, J. Rushby, J. Wing, and P. Zave. "An Invitation to Formal Methods," *IEEE Computer* (29:4), April 1996, pp. 16-30.

[Santhanam 94] Santhanam, R. and M. Sein. "Improving End-user Proficiency: Effects of Conceptual Training and Nature of Interaction," *Information Systems Research*, (5:4), December 1994, pp. 378-399.

[Schneider 83] Schneider, B. Interactional Psychology and Organizational Behavior," *Research in Organizational Behavior*, (5), 1983, pp. 1-31.

[Tait 88] Tait, P. and I. Vessey. "The Effect of User Involvement on System Success: A Contingency Approach," *MIS Quarterly* (12:1), March 1988, pp. 91-107.

[Taylor 95] Taylor, S. and P. Todd. "Understanding Information Technology Usage: A Test of Competing Models," *Information Systems Research*, (6:2), June 1995, pp. 144-176.

[Tetrick 87] Tetrick, L. and J. LaRocco. "Understanding, Prediction, and Control as Moderators of the Relationships between Perceived Stress, Satisfaction, and Psychological Well-Being," *Journal of Applied Psychology*, (72:4), 1987, pp. 538-543.

[Tornatzky 90] Tornatzky, L. and M. Fleischer. *The Process of Technological Innovation*, Lexington, Mass: Lexington Books, 1990.

[Vessey 86] Vessey, I. "On Program Development Effort and Productivity," *Information and Management*, (10), 1986, pp. 255-266.

[Vessey 98] Vessey, I. and R. Glass. "Strong vs. Weak Approaches to Systems Development," *Communications of the ACM*, (41:4), April 1998, pp. 99-102.

[Walston 77] Walston, C. and C. Felix. "A Method of Programming and Measurement Estimation," *IBM Systems Journal*, (16), 1977, pp. 389-407.

[Wing 90] Wing, J. "A Specifier's Introduction to Formal Methods," *IEEE Computer*, (23:9), September 1990, pp. 8-24.

[Wynekoop 92] Wynekoop, J., J. Senn, and S. Conger. "The Implementation of CASE Tools: An Innovation Diffusion Approach," in K. Kendall et al. (Eds.), *The Impact of Computer-Supported Technologies on Information Systems Development*, Minneapolis, MN: Elsevier Science Publishers, B.V., 1992, pp. 25-41.

[Yourdon 96] Yourdon, E. *Rise and Resurrection of the American Programmer*, Yourdon Press, Prentice-Hall PTR, 1996.

Appendix A: Questionnaire

CONSENT FORM

You are invited to participate in a study to analyze the use and impacts of the Personal Software Process (PSP). Because you are a current or former user of PSP, we are interested in your feedback regarding your use of PSP and its impact on your software development activities. If you agree to participate in this study, you will be asked to complete an 8-page questionnaire, which takes approximately 30 minutes to complete.

All information collected in this study will be kept strictly confidential, and your name will not be associated with any paper materials. To ensure confidentiality of responses, this form will be separated from the questionnaire prior to any data analysis. In any presentations or reports that may be based on this questionnaire, no one will be identified or identifiable, and only aggregated data will be presented. The information collected in this questionnaire will not be disclosed to anyone without your written permission, nor will it be used by your management for performance purposes.

If you have any questions about the study or research subjects' rights, please contact Dr. Alan Hevner (813-974-6753) or Dr. Gina Green (254-710-6210). Thank you.

You are making a decision about whether or not to participate in this study. Your signature indicates that you have read the information provided above and have decided to participate. You may withdraw from the study at any time after signing this form.

It is specifically understood that in any publication, I will not be identified by name or any other personally identifying information.

Signature _____ Date _____

There are eight sections in this survey. Please respond to all questions in Sections I through VII. If you are receiving this survey via email, please use bold type and/or underlining of responses in place of circling of responses. Section VIII provides space for comments--this section is optional. The approximate time to complete the survey is 30 minutes. When you have completed the survey, please email the survey to *Gina_Green@baylor.edu*, or mail to us at the following address:

Dr. Gina Green
Information Systems Department
Hankamer School of Business
Baylor University
P.O. Box 98005
Waco, TX 76798-8005

Please return completed survey to us within 30 days. Thank you for your participation in our study.

Section I. General Information

Please answer the following background questions:

1. How long have you worked in the Information Systems field? _____ years

2. Circle the name that most closely matches your current position (circle only ONE):
   a. programmer
   b. analyst
   c. designer
   d. project manager
   e. manager

3. Circle the highest degree that you have completed:
   a. high school diploma
   b. bachelors
   c. masters
   d. doctorate

4. Age: _____ years

5. Gender (circle ONE):
   a. male
   b. female

6. On how many projects have you used PSP techniques? _____ projects

7. For how long have you used PSP techniques? _____ years and _____ months

8. Please list techniques, methods, and tools that you have used or are currently using with PSP. Examples would include object-oriented design, task analysis, statistical testing, CASE tools, rapid application development, etc.

_____

_____

_____

_____

Section II. Involvement

The following questions relate to your involvement in the decision to use the Personal Software Process (PSP) in software development. Please read each question carefully and circle on the scale to the right of each question (see definitions below) the choice that best represents your situation.

to no extent	to a very little extent	to a little extent	to some extent	to a large extent	to a very large extent	to a great extent
NO	VLT	LT	SM	LG	VLG	GR

To what extent did you...

1. ...lead the effort to adopt PSP?..... NO VLT LT SM LG VLG GR
2. ...have responsibility for requesting resources to support the use of PSP?..... NO VLT LT SM LG VLG GR
3. ...have responsibility for estimating costs or benefits of implementing PSP?..... NO VLT LT SM LG VLG GR

How much did you participate in...

4. ...determining the objectives of PSP use? ..... NO VLT LT SM LG VLG GR
5. ...assessing alternative ways of meeting your development needs?..... NO VLT LT SM LG VLG GR
6. ...identifying sources of information on PSP?..... NO VLT LT SM LG VLG GR
7. ...initiating the effort to adopt PSP?..... NO VLT LT SM LG VLG GR
8. ...determining which development tools and techniques can address your development needs?..... NO VLT LT SM LG VLG GR

Section III. Organizational Environment

The following questions relate to support of PSP by management, and your organizational climate in general.

1. An Information Technology (IT) "champion" can be defined as one who "actively and vigorously promotes their personal vision for using IT, pushing [a] project over or around approval and implementation hurdles (Beath, 1991)." Keeping this definition in mind:

   ... to what extent was there a "champion" for PSP in your organization?..... NO VLT LT SM LG VLG GR

2. If you feel there was a PSP champion, please identify the title and department of this champion:

   TITLE _____ DEPARTMENT _____

For the following questions, please circle on the scale to the right of each question the choice that best represents your situation.

to no extent	to a very little extent	to a little extent	to some extent	to a large extent	to a very large extent	to a great extent
NO	VLT	LT	SM	LG	VLG	GR

3. If you feel there was a PSP champion, how effective was the champion in promoting the use of PSP? ..... NO VLT LT SM LG VLG GR

4. How supportive was top, non-IS management in the selection and implementation of PSP? ..... NO VLT LT SM LG VLG GR

5. How supportive was IS management in the selection and implementation of PSP? ..... NO VLT LT SM LG VLG GR

Please circle on the scale at the right of each question the choice that best represents your belief.

Strongly Disagree	Disagree Slightly	Disagree	Neutral	Agree Slightly	Agree	Strongly Agree
SD	D	DS	N	AS	A	SA

6. Management provided me with time for training that I needed in order to use PSP effectively. ..... SD D DS N AS A SA

7. Management provided me with funding for training that I needed in order to use PSP effectively. ..... SD D DS N AS A SA

8. I had easy access to people of the necessary expertise to help me make the transition to PSP. ..... SD D DS N AS A SA

9. For making the transition to PSP, I felt I had a solid "network of support" in the form of knowledgeable colleagues, internal support personnel, and/or outside consultants. ..... SD D DS N AS A SA

10. My superiors expect me to use PSP ..... SD D DS N AS A SA

11. My use of a PSP is voluntary (as opposed to required by my superiors or job description). ..... SD D DS N AS A SA

12. My supervisor does not require me to use PSP ..... SD D DS N AS A SA

13. Although it might be helpful, using PSP is certainly not compulsory in my job. ..... SD D DS N AS A SA

14. Use of PSP is a part of my performance plan and/or job description. ..... SD D DS N AS A SA

15. There was a pre-specified sequence of steps that I was required to follow in my use of PSP. ..... SD D DS N AS A SA

16. I was required to follow existing standards in my use of PSP ..... SD D DS N AS A SA

Section IV. Training

The following questions relate to training you received on PSP concepts. If you received any form of training on PSP, please read each question carefully and circle on the scale at the right the choice that best represents your situation or beliefs. If you did not receive any training on PSP, please skip to section V.

Strongly Disagree	Disagree Slightly	Disagree	Neutral	Agree Slightly	Agree	Strongly Agree
SD	D	DS	N	AS	A	SA

1. The training I received on PSP was adequate ..... SD D DS N AS A SA
2. The quality of training I received on PSP was high ..... SD D DS N AS A SA
3. There was not enough training for me on how to use PSP..... SD D DS N AS A SA
4. I have received the training I need to be able to use PSP effectively ..... SD D DS N AS A SA
5. Training on PSP was received at a time that was appropriate to ..... SD D DS N AS A SA
   when I began to use PSP.
6. Training on PSP was received at a time when I was ..... SD D DS N AS A SA
   simultaneously working on other assignments.
7. Where did you receive your PSP training? _____

Section V. Assessment of PSP Characteristics

The following questions relate to your assessment of PSP based on your use of PSP techniques. Please read each question carefully and circle on the scale at the right the choice that best represents your belief.

Strongly Disagree	Disagree Slightly	Disagree	Neutral	Agree Slightly	Agree	Strongly Agree
SD	D	DS	N	AS	A	SA

1. I found it easy to use PSP..... SD D DS N AS A SA
2. The tools and techniques of PSP were clear and understandable..... SD D DS N AS A SA
3. I found PSP to be flexible to implement..... SD D DS N AS A SA
4. It was easy for me to become skillful at using PSP..... SD D DS N AS A SA
5. Using PSP improved my job performance..... SD D DS N AS A SA
6. I feel that use of PSP was successful in transforming me into a ..... SD D DS N AS A SA
   more disciplined software engineer.
7. Using PSP made it easier to do my job..... SD D DS N AS A SA
8. I found PSP useful in my job..... SD D DS N AS A SA
9. The information provided by PSP provides sufficient information..... SD D DS N AS A SA
   for improving software development effectiveness.
10. PSP provides information that is exactly what I need..... SD D DS N AS A SA
11. PSP meets my software development needs..... SD D DS N AS A SA

Section VI. Impacts of PSP

The following questions relate to your assessment of the impacts of PSP techniques on your work. Please read each question carefully and circle on the scale to the right of each question the choice that best represents your beliefs.

to no extent	to a very little extent	to a little extent	to some extent	to a large extent	to a very large extent	to a great extent
NO	VLT	LT	SM	LG	VLG	GR

1. Since use of PSP, to what extent do unexpected results occur during systems development? NO VLT LT SM LG VLG GR

2. To what extent does the use of PSP techniques allow you to better predict the effort required for software development? NO VLT LT SM LG VLG GR

3. To what extent does the use of PSP techniques allow you to better predict the quality of software that you develop? NO VLT LT SM LG VLG GR

4. To what extent can you decide what parts of PSP you will use? NO VLT LT SM LG VLG GR

5. To what extent can you decide when you will use PSP techniques? NO VLT LT SM LG VLG GR

6. To what extent does the use of PSP allow you the opportunity for creativity in systems development? NO VLT LT SM LG VLG GR

Please circle on the scale at the right of each question the choice that best represents your belief.

Strongly Disagree	Disagree Slightly	Disagree	Neutral	Agree Slightly	Agree	Strongly Agree
SD	D	DS	N	AS	A	SA

7. Use of PSP has enhanced the functionality of applications that I build SD D DS N AS A SA

8. Use of PSP has decreased the number of errors in software products I build SD D DS N AS A SA

9. Use of PSP has improved the quality of software products I build SD D DS N AS A SA

10. Software developed with PSP requires less maintenance SD D DS N AS A SA

11. Use of PSP has significantly improved my documentation of software products SD D DS N AS A SA

12. Use of PSP has made me more conscious of software quality SD D DS N AS A SA

13. Use of PSP has greatly speeded up my development of new applications SD D DS N AS A SA

14. Use of PSP has definitely made me more productive SD D DS N AS A SA

15. Use of PSP has significantly reduced the time I spend in software development. SD D DS N AS A SA

Please answer the following questions by circling the applicable answer:

16. On average, approximately what percentage improvement in productivity, if any, have you experienced since implementing PSP techniques (circle only one)?

    1. none 2. 1-25% 3. 26-50% 4. 51-75% 5. >75%

17. On average, approximately what percentage reduction in software errors, if any, have you experienced since implementing PSP techniques (circle only one)?

    1. none 2. 1-25% 3. 26-50% 4. 51-75% 5. >75%

Section VII. Individual Use of PSP

The following questions relate to your experience with learning PSP techniques, and applying the techniques to your software development efforts. Please read each question carefully and circle on the scale beneath the question the choice that best represents your situation.

1. The proportion of projects I use PSP with (circle only one):

   1. none 2. 1-25% 3. 26-50% 4. 51-75% 5. >75%

2. My use of PSP was/is (circle only one):

   1 = in initial project only
   2 = in mostly small projects, but not in large projects
   3 = in a mixture of small and large projects
   4 = in mostly large projects, but not in small projects
   5 = completely routine (in all my projects)

3. My use of PSP was/is (circle only one):

   1 = mostly critical projects
   2 = mostly non-critical projects
   3 = a mixture of critical and non-critical projects

4. The phases that I typically use PSP principles in are (circle all that apply):

   1. Requirements specification
   2 . Software analysis
   3. Software design
   4. Implementation
   5. Testing
   6. Maintenance

5. How often do you use PSP?

Extremely infrequent	quite infrequent	infrequent	sometimes	frequent	quite frequent	extremely frequent

Please circle on the scale at the right the choice that best represents your belief.

Strongly Disagree	Disagree Slightly	Disagree	Neutral	Agree Slightly	Agree	Strongly Agree
SD	D	DS	N	AS	A	SA

6. Initially, I felt that there existed a large gap between my existing skills and knowledge and those required by PSP. ..... SD D DS N AS A SA

7. Major modification in our software development policies and procedures was necessary for PSP to truly fit in. ..... SD D DS N AS A SA

8. The "language" of the PSP approach was not familiar to me ..... SD D DS N AS A SA

9. The underlying methodology of PSP was different from what we had used before. ..... SD D DS N AS A SA

Please use this page to provide us with any additional thoughts you may have regarding the use of PSP in your software development environment. These comments may help us better understand your answers, or may provide us with future questions that need to be addressed in research. Again, we would like to thank you for your participation in this survey.

Appendix B: Questionnaire Response Material Concerning PSP

The following questionnaire response material, which specifically concerns the Personal Software Process approach, was collected by the authors for a presentation at the Software Engineering Institute.

PSP Use by Frequency

extremely infrequent	4%
quite infrequent	8%
infrequent	8%
sometimes	18%
frequently	21%
quite frequent	26%
extremely frequent	15%

PSP Use by Project Percentage

Use of PSP was 0%	10%
Use of PSP was 25%	26%
Use of PSP was 50%	13%
Use of PSP was 75%	14%
Use of PSP was >75%	37%

PSP Use by Project Phase

requirements	32%
analysis	40%
design	80%
implementation	92%
testing	65%
maintenance	24%

PSP Use by Project Size

pilot only	31%
small only	10%
mixture	28%
large only	7%
routine	24%

PSP Use by Project Type

critical	36%
non-critical	25%
mix	39%

Challenges of PSP Use: Summarized Comments

- Lack of immediate feedback on improvements
- Productivity
- Estimating
- Difficult to implement in teams
- Not easily adapted to maintenance and front-end activities (analysis, database design)
- Language-specific
- Not easily used for 4GLs, OOP, visual programming
- Lack of tool support to ease data gathering and analysis
- Lack of allowance in project schedules

Benefits of PSP Use: Summarized Comments

- Improved quality and productivity anticipated
- Gives structure to programming tasks
  - "It gives me the structure I need to come back to programming tasks and pick them up and start up again with some hope that I can maintain a high level of quality throughout."
- PSP is worth the training even if not used
  - "Most of my work is experimental, so I don't strictly use it; however, *how* I do things was influenced by the techniques in the class. I understand the flow of my work better and know where to look for trouble."
  - "The primary benefit of PSP is that it raises the level of consciousness of the developer. ...I've discarded most of the methods [taught in class] and am defining my own measures of improvement tailored to fit the way I work. ...The *principles* underlying the PSP are very valuable."
  - "The *pedagogical PSP* taught in the discipline for software engineering is inappropriate for industrial software development. The *underlying concept*—empirically guided software process improvement—is enormously useful."

Additional Comments from PSP Survey Respondents

In addition to the comments presented above, respondents to the PSP survey made the following comments:

The problems we are encountering are as follows:

- Starting to use PSP with a new language may produce questionable results. Often design decisions are made during code phase out of necessity.
- Time and budget constraints make it difficult to adopt PSP. There is pressure from the customer to produce better, faster, cheaper code. Note: the customer, not management, is exerting the pressure.

These 2 factors make PSP difficult to implement. It is hard for the customer to understand "Trust me, you'll be glad I used PSP." It is difficult to have a clear-cut design and code phase when you are quite unsure of the language capability. That is coupled with the fact that the tool and language are so new that they contain bugs.

The difficulty I have had with applying PSP has been trying to keep pace with changing platforms and languages and trouble adapting PSP to a maintenance environment.

1. An all-encompassing software needs to be produced for the entire PSP (data collection, statistics, analysis, graphs ... everything).
2. PSP should initially be included in project schedules until that time becomes insignificant (plan, design reviews, post-mortem).
3. More study should be done on how to use PSP with new tools/environments...Is PSP useful when learning a new tool?
4. Companies using PSP should have a full-time PSP resource person.
5. Encourage participation by enforcing biweekly reports to management/peers.
6. Management should keep workforce informed on any/all PSP decisions/actions.
7. Include PSP in state university, programming-related curriculum.

As project lead, I used PSP for tracking my project lead activities.

Our company provided us with the time for the course but not for the time required to do the homework exercises. This proved to be a larger effort than anyone expected and resulted in long hours of our personal time.

I agree with the benefits of the PSP techniques on an individual level but I have found it difficult to implement on a team level since not everyone on the team has been PSP trained. I sometimes also feel my management is more concerned with collecting PSP data than actually getting the deadlines met.

Overall the positives of PSP outweigh the negatives. We have a better knowledge of where we are on projects and what resources we require. Sometimes the overhead of taking the PSP data is a burden but with automating the process this should get better. Since this is my first project using PSP it is hard to gauge all of the benefits versus the negatives. In time these will become clearer.

PSP/TSP was used at the beginning of my software project. But, we have not yet completed the entire software lifecycle. Matter of fact, we are just starting the implementation phase. We plan to gather defect data throughout the remaining lifecycle; included non-PSP teammates. So far, PSP/TSP has been a valuable tool in project planning (conceptual design, estimating, and developing a schedule). During the specification and design and test procedures, PSP is only used to gather time, update schedules, calculate earned value, and report to management.

I volunteered to take the PSP course. I thought PSP would make me a better programmer. Whether or not PSP has made me a better programmer or there was just a learning curve to overcome, I am a better programmer today than I was [before the course]. PSP has taught me discipline in developing software. I compile error free >90% of the time. I think PSP has definitely made me more aware of typical syntax and data type errors I commonly made in the past, and I am more observant while coding so that I don't make these errors. PSP has also made me more confident in estimations of time to complete an assignment, and [has provided] justification for why the assignment is "taking so long". I have data to back up my words—this is critical in our current job environment.

Applying PSP to projects is enforced by my boss, but I would apply parts of it even if [it were not] enforced. In some areas, PSP provides me with more data than I feel I need. As time passes, and projects are completed, some of the PSP techniques could possibly be removed at a personal level. My boss was very flexible as to how we applied PSP to the project level, which eased the effort in applying PSP to the project. Without flexibility, it would have been a nightmare.

I feel the personal use of PSP is very beneficial. I have not worked directly with software development since the PSP class. I am working on applying PSP principles to database development. I am confident that PSP will help us provide a better quality product.

PSP at an individual level has really been beneficial in allowing me to become a better software engineer and in understanding the *software* development lifecycle. It is very hard, though, to transition PSP into a team project, especially if

- Everyone is *not* PSP trained
- Majority is against PSP concept
- Team process scripts have not been developed prior to starting of project
- No automated way of collecting and analyzing data has been done.

But, PSP for a team project, I found very beneficial, even with the above obstacles., The task and schedule planning sheets really gave the team insight into exactly what needed to be done and when, and also allowed the team to foresee resource problems. Overall, I think PSP is very beneficial in predicting future estimates and is a valuable resource.

What I gained most from PSP was more effective techniques for reviewing (for design and code) my work. Through PSP I realized the importance of following a checklist of review items. I was able to review my metrics and come up with an optimal and organized method of

reviewing my work. However, having only used PSP for the coursework and for a short time since, I have not noticed a significant amount of time saved. Also, I realize that it is going to take quite awhile to come up with enough metrics to produce accurate estimations, possibly years. So it is quite frustrating to think that it will be some time before it is possible to see if PSP is really improving my productivity and estimating abilities.

My type of work changed just as I began learning how to use PSP. I was primarily a developer, but now do mostly support, debugging, and third-party project management. It would be helpful to see how PSP techniques could be adapted and used in support/debugging, if that's possible.

By the time I finished the PSP course

- we no longer did [new software] development
- I had no time to work on an adaptation [of PSP to a non-software-development environment]

I am [now] in software maintenance and hope to get back to PSP at some future date.

For a real business project, many languages and software packages are involved and many software developers are grouped together to fulfill the task. Unfortunately, most of top-level management doesn't care about PSP, he/she defines a due date, and developers have to finish the project before the day. For a "non-smart" programmer, no matter how well he/she handles PSP, he/she won't build a high-quality product.

My participation in PSP was required. I had been doing PSP-like practices of my own before the pilot, so the formalized use of PSP was not much of a stretch for me.

It was difficult to judge or measure the influence and effectiveness of PSP in our circumstances. We currently have no means to measure its effectiveness against our current process as we do not use PSP for development.

PSP should be a methodical way to get answers *that can't be gotten adequately by simpler methods*. If all the overhead is applied by mandate to trivial questions or ones where the answer can be guessed with high confidence then PSP will look bad (cost/benefit). There need to be specific problems/questions identified where PSP is the right tool for the job —e.g., if we identify a repeating process with inefficiencies, and determine an improvement with *obvious* benefits, it does not require the full PSP process to continue. PSP should be a tool for getting *important* answers that can't be determined less formally.

With regard to my PSP training, the only major problem I had was the class size was too large for one instructor to provide adequate time for questions and assignments. This problem was very evident to the class and instructor.

I have finished one project using PSP techniques and I have just started working on another project using PSP techniques. I have no data of productivity before working on these projects, so I cannot [assess quality and productivity] accurately. But I am sure that my quality of work has increased a lot after using PSP.

People have to be cautious in their use of PSP when using data from a prior project to start on a new project. There are startup overheads associated in any project (e.g., training) which are generally not identified and isolated. This results in lower productivity in the beginning. The models used for estimation should take this into consideration if the startup tasks cannot be identified.

We regularly use PSP. I have found PSP useful in planning—particularly in those projects where I am the only team member and I am responsible for project management. It has also been helpful in achieving high-quality products. However, it is hard for me to gauge the effect PSP has had to "before," because I have only worked on one software development project previous to using PSP. It is also hard to gauge "improvement" since PSP has been introduced because each project is usually vastly different in characteristics from the next: different language, platform, etc. I have worked on other consultative projects where my use of PSP has been limited to project planning.

As an "on-site" programmer it is tough for me to convince my superior (on-site) why I am using PSP. Now they understand the importance of PSP (of course only when I meet my deadlines). Only drawback for me using PSP was "documentation" when I started using PSP. But along the way I got used to it.

As a project manager, PSP helps me a great deal with estimating, tracking, and understanding the quality of the product being developed. For us, some challenges we have not completely overcome involve

- the use of PSP when functionality and plans change (during a PSP cycle) due to external circumstances
- use of PSP during Maintenance phases of a project
- use of PSP with 4GLs.

Originally, I worked in [an] office which mandated the use of PSP. The project that I worked on at that time was ideal for PSP, because it was new development. Later, I moved to [another] location with a more ad-hoc approach to software development. My current location does not support PSP. Also, I'm not usually asked to estimate time, and when I am, I can usually use my intuition to accurately estimate such small projects. Yet, I have been collecting data and I plan to use PSP in the coming year for the sake of reducing cycle time and defects.

I did not use all the techniques of PSP in my projects; I just tailored some concepts of PSP to my project. Before I was introduced to PSP, I was already a process-oriented and disciplined person, so I didn't personally see marked benefits in terms of productivity and time savings. The current PSP works very well for small projects but not for medium to large projects; that's why some tailoring was required. On the whole, the PSP concepts, rather than the actual way of implementing PSP, are very good.; PSP should be tailored to suit one's project.

PSP has helped me tremendously in software development. It was an eye-opening experience. I can better manage my development skills so I can produce high-quality, less expensive products.

While I know that using PSP will make me a better developer, I still haven't used it for a real project. This is due in large part to an organizational culture that hasn't integrated PSP—our

group procedures don't do anything to encourage individuals to use PSP and I haven't been able to motivate myself to use PSP on my own.

It is my experience that PSP data and techniques are useful when you are working on projects where a large percentage of what needs to be done is known and understood. In developing for new platforms or unfamiliar products, I found that PSP estimates were off by as much as a factor of 4. Now, one might argue that this will improve with time. Well, that's true as long as things don't change. In our environment, we are constantly doing new things, making the gathering of enough PSP data to be useful difficult.

Two things have contributed to making the transition from classroom to real life awkward. First, the tools used in class to automate data-gathering and analysis were *Windows* based but most of [our] machines run *Unix*. Second, while the course exercises called for writing code from scratch, and reusing components, my work requires more modifying existing code than writing original code.

I am very positive about the benefits of PSP and the increased quality and productivity when using PSP. But, since taking the course, I have spent the majority of my time doing high-level project and product definition and management, and very little time performing the design, code, test portions of the PSP cycle; thus I have very little predictive data for PSP purposes and few actual results. However, I do continue to use PSP in my own, personal projects and have found it beneficial there.

I used PSP on one small project. It did not improve my productivity or reduce my error rate (meaningful errors). There is no requirement or encouragement to use it. I don't strictly use [PSP]; however, how I do things was influenced by the techniques in the class. I understand the flow of my "work" better and know where to look for trouble. I can head off trouble before it starts. So it was worth having the training, even if it is not formally used here.

The primary benefit of PSP is that it raises the level of consciousness of the developer. PSP encourages the developer to look over his own shoulder, so to speak. Common mistakes and process improvements are frequently obvious and simple once identified. I personally found the PSP as presented in our training class to be inappropriate for me as an individual and also

with respect to the goals set by our management. I found using PSP as presented nearly impossible in daily life. Accordingly, I've discarded most of the methods and attempted to integrate the principles of PSP into my work in other ways. This exercise has included defining my own measures of improvement and developing a set of tools tailored to fit the way I work. Since I first encountered PSP I've observed it turning into a bit of dogma. Its presentation and packaging is rigid and excessive thus obscuring the principles behind PSP's clumsy worksheets, forms, and measurements. (Your questions regarding the required use of the PSP are a perfect example of forgetting that PSP is a *personal* process). In summary, the principles underlying PSP are very valuable, the presentation and packaging, however, make PSP look like just another managerial boondoggle.

We are halfway through detailed design on our first PSP project so the jury is still out. We are also trying to implement "TSP." Much of the effort seems cumbersome but will no doubt get better with time and experience.

The development we are working on is in the design phase. As a condition of the work currently in progress, it was a group consensus to use PSP and stick to it. Our management encouraged its usage but said we would have to follow through with it if we committed to it. The group voted and a majority voted to use PSP. I am not directly performing design work, but I have had a greater appreciation of the work captured for each phase of development. I feel the team has a better grasp of what is required for this job more thoroughly at the design phase than in any previous development.

Since PSP is not widely used, you will find a lot of "n/a's" for awhile. Although not PSP specifically, I received a lot of methodology, inspection, etc., training before coming to [my current company].

To be honest, we are only in detailed design of our first project using PSP. Shortly before the start of detailed design, TSP [was introduced] to our team and we are following through with it.

PSP slows software development to an extent that management is willing to disregard it in favor of traditional "hacking," which at least produces visible results quickly.

PSP does not as readily scale up as alluded to. More work needs to be done here.

PSP is very sensitive to platform and language used. Methods to decouple the PSP should be explored further.

[PSP] books are difficult to reference in real practice. More authors/books are needed to specialize efforts versus convincing the trainee to use the PSP.

Avoiding software "tools" is essential to understanding the true meaning of PSP. Programmers *immediately* want to buy tools and automate everything in lieu of understanding PSP.

PSP made me think about process beside the software product. I see many benefits in using PSP for myself; but I doubt that people like it: it needs a lot of discipline.

The pedagogical PSP taught in the discipline for software engineering is inappropriate for industrial software development. The underlying concept—empirically guided software process improvement—is enormously useful. The SE/PSP community has yet to fully understand this distinction.

Tool support for PSP is essential. The overhead in manual data collection and analysis is stupefying.

PSP helps with organizing a management job also.

The application of PSP in a research setting (like at a university or other) seems to be not as promising as in industry. This might be due to the fact that no two software projects are of the same requirements, it is a very explorative work setting (as opposed to repeated requirements), and that software development occupies only part of [research].

As a Manager I do not do hands-on work in realizing the product. However, occasionally I used the PSP principles when giving presentations, keeping meetings and minutes, etc.

Colleagues of mine typically develop products. However, more and more they are active in the other areas, like maintenance and support. Though the PSP principles are usable, in practice this leads to a weak usage of PSP. This [is due to] simple reasons. The line counting program does not work to determine added and deleted lines of code so that has to be done manually. It is very difficult to predict effort of a change to code that is not yours, i.e., there must be much initial study prior to making a change. This results in large differences in the estimations and results database.

I've found the intro of PSP at my company a painful process. In the initial information session, [many] engineers were present. [Now] I am the only engineer who is using PSP (or what is now an adaptation of it) on a regular and continuous basis. Learning and using PSP is a real commitment. It is not clear how to convince engineers to get seriously involved.

The initial champion dropped the effort. I became a champion myself. Nobody is asking me to use PSP but I must admit that I cannot leave it at this point.

Training was excellent. Publications are thorough, it is easy to communicate with peers (outside my company). It is also very easy to keep up to date (Team PSP, emails, etc). Several articles appear in software engineering publications and my motivation is extremely high. It would be much nicer if I could exchange practical application problems with peers in my company. Tools are poor (as far as I know). I have developed my own tools by producing software that made the use of PSP a real charm. I wish that the tools would be much less manual. Timekeeping, for example, can be a pain in the neck if you have no means to quickly record task changes. I solved the problem by introducing a *time recorder*. The learning process is long and complex, and the learning effort (with the exercises) discourages engineers. The benefits are not clearly demonstrated, and even if they are, engineers usually view the use of PSP as an additional burden. I have discovered that there is a loss of productivity at the beginning of the usage in real projects (after the training). This can discourage engineers to continue or improve the personal process. One could wonder if some personal characteristics could influence the adoption of PSP. Being methodic, rational, objective, having a sense (or a will) of organization. These are several characteristics that I discovered in those that I know who continued using PSP. Those who did not continue often displayed a more *artistic* view of software engineering. Further research on this subject could contribute to identifying who would or would not possibly adopt PSP. PSP is an excellent tool, no doubt about that.

The positive impact is doubtless. I could not go back to the way I worked before using PSP. My productivity is much higher. The quality did not improve in the same proportion (even if it is significant), but it certainly helps me keep excellent track of defects at all levels. PSP is an essential mirror that measures my professional efficiency at all levels. On the negative side, the use of PSP [can] cause several sources of frustration. The fact that [one is] using a personal process and [team members] don't (at least not systematically) often causes some sources of dissention. They do not understand the process and the benefits. In teams, this can have a frustrating impact. Team PSP will probably address that issue.

Impacts [of PSP] have transcended my professional life, and have some ramifications in my personal life (learning activities, hobbies, etc.). [Watts] Humphrey was right when he said that it is a way of life. PSP doesn't prevent creativity but, on the contrary, can help accomplish quality realizations. My creativity has more impact since it can be planned, prepared, and executed instead of just stating fuzzy concepts and ideas. *Software* engineering is mainly distributed in small- and medium-sized corporations and I truly believe that this is where PSP has the most problems.

We are currently adopting an adapted version of PSP in the direction of Team Software Process. The adoption was made by ourselves. We are still going on with our PSP training.

My organization has its own software process which is compatible with PSP. What I use from PSP is creation of automated techniques for tasks I do frequently and time estimation on the scale of less than one day, days, weeks, or months. I don't use lines of code estimation or tracking. I do go through an informal design review for every project and if a particular section of code is complex I will do a code walkthrough before testing. I have found that significantly reduces bugs. I do no checking before compilation as I view the compiler as being an automated syntax checker.

The [PSP] techniques are clear and understandable, but one of my main difficulties using PSP at work has been the lack of tools. The tools I used in class (some Excel spreadsheets that the instructor provided) were pretty good, but I have not had time to adapt them to my work or to see if any other tools are available.

I have been using PSP on my own. I'm still on the learning curve and have had no formal training. The information is easy to read from the books available, but sometimes hard to implement due to lack of self-discipline. The company I work for supports my actions, but does not provide any formal training.

I find that using PSP principles helps me immensely in estimating both personal and project work. However, I find that the organization as a whole will not use PSP unless there is a good reason to use it. The only project that I am aware of that is currently utilizing the PSP techniques is one that has employed the Team Software Process. Unless a team is committed enough to use TSP, I do not believe that most people on the team will use PSP, even if they have been trained in it. We have trained several people in our organization [but] only [a few] people are using it as a part of their project work. Those [few] are involved in the TSP. I believe that true benefits from the PSP require

- full management awareness and support (i.e., the managers should be using PSP techniques themselves)
- a committed team (i.e., a TSP-trained team)
- positive motivation for PSP Use.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (LEAVE BLANK)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	April 1999	Final

4. TITLE AND SUBTITLE
Perceived Control of Software Developers and Its Impact on the Successful Diffusion of Information Technology

5. FUNDING NUMBERS
C — F19628-95-C-0003

6. author(s)
Gina Green, Baylor University
Alan R. Hevner, University of South Florida

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

8. performing organization report number
CMU/SEI-98-SR-013

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12.A DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified/Unlimited, DTIC, NTIS

12.B DISTRIBUTION CODE

13. ABSTRACT (MAXIMUM 200 WORDS)

Why are beneficial software engineering practices not being used effectively in the development of software systems? This question has intrigued researchers in software engineering for many years [Parnas 85]. Billions of dollars per year are spent, and a large proportion wasted, on building and maintaining software systems that are either never completed or, if completed, are of poor quality. This state of software development has led to the introduction of innovative tools and techniques to support the software development process. Initial evidence from use of these tools and techniques shows significant improvements in development productivity and software quality. However, many of these potentially beneficial tools and techniques have not been widely adopted or diffused. This research seeks to examine the reason for *why* this is so: What factors explain the successful diffusion of new software development techniques into practice?

Software development techniques are viewed as a subset of the broader category of information technology (IT). A research framework is developed that explains the complex relationship between developer involvement in the IT adoption process, characteristics of the environment into which the IT is introduced, and IT diffusion success. The framework posits that the effects of developer involvement and IT diffusion environment characteristics on IT diffusion success are mediated by (1) the developer's perceived control over his work when using the IT, (2) the developer's perceptions of the IT, and (3) the developer's perceptions of the impacts of IT use. Using the Personal Software Process℠ (PSP℠) approach as an example of an innovative IT, we develop a survey instrument for the collection of study data, which are analyzed using path analysis. The survey was distributed to software developers who have used PSP on software development projects in industry. Results find support for mediating effects of Perceived Control on the relationships between Developer Involvement, Training, Voluntariness, Champion Support, and IT Diffusion Success. Results also find direct impacts of Developer Involvement, Training, and Voluntariness on IT Diffusion Success.

This research contributes to research by providing a model that integrates knowledge from the research fields of information systems (IS), diffusion of innovations, software engineering, and social psychology to better understand why user involvement and diffusion environment impact diffusion success. Further, it provides a measure for perceived control in an IS context. This research contributes to practice by stressing the importance of software developer perceptions in determining IT diffusion success. The research underscores the importance of creating a perception of greater control over the software development process in order to positively influence diffusion success. Guidance is provided to software development practice by stressing the importance of software developer involvement, training, voluntariness, and champion support in the introduction of new IT. These variables can be influenced by management to enhance perceived control and, therefore, better ensure successful diffusion of new software development practices in organizations.

Editor's Note: This SEI Special Report also includes information from a presentation delivered by the authors at the Software Engineering Institute. The information, which appears in Appendix B, is material collected by the authors from their questionnaire and specifically concerns the Personal Software Process℠ (PSP℠).

14. SUBJECT TERMS
SOFTWARE DEVELOPMENT PROCESS, INFORMATION SYSTEMS, DIFFUSION OF INNOVATIONS, SOFTWARE ENGINEERING, SOCIAL PSYCHOLOGY, PERCEIVED CONTROL, PERSONAL SOFTWARE PROCESS, DEVELOPER INVOLVEMENT, TRAINING, VOLUNTARINESS, CHAMPION SUPPORT, IT DIFFUSION SUCCESS

15. NUMBER OF PAGES
130

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102